Universidad Carlos III de Madrid
Escuela Politécnica Superior

Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF

Proyecto de Fin de Carrera

Ingeniería en Informática

Autor: Javier de la Blanca Teba
Tutor: Fernando Paniagua Martín

Marzo 2009

A mis padres y mi hermano

Agradecimientos

Quiero aprovechar este importante momento en mi vida para agradecer a todo el mundo que creyó en mí desde el principio, fundamentalmente a mis padres, hermano, abuelos, familia y amigos.

Brindo este proyecto de fin de carrera, consecuencia y fruto de unos maravillosos años en la universidad, a todas las personas que de una u otra forma han colaborado con él, facilitándome enormemente la tarea. A mis padres y mi hermano por aguantarme en momentos difíciles y a mis compañeros de camino, que ha habido muchos y muy buenos. También va dirigido, cómo no, a dos importantes personas de mi recta final en la universidad, Fernando Paniagua, mi tutor del proyecto, y Ricardo Colomo, las cuales me han apoyado muchísimo.

No quiero nombrar a nadie, porque sois muchas las personas importantes en mi vida, y no querría dejar a alguien fuera sin merecerlo, ¡gracias a todos!

Resumen

La complejidad de las aplicaciones informáticas se vuelve cada vez más elevada con el paso del tiempo con la llegada de nuevas y más potentes tecnologías. Es por ello que se deben proporcionar mecanismos de comunicación con el usuario más útiles y acordes con la lógica de negocio, además de intuitivos de tal forma que a través de los mismos se puedan llevar a cabo todas las, en muchos casos, complejas funcionalidades.

Como consecuencia de lo mencionado, el actual enfoque en la creación de aplicaciones es el de dotar a las mismas de interfaces de usuario muy potentes y extremadamente complejas, con gráficos en dos y tres dimensiones e inclusive con contenido multimedia, que aporten un alto grado de interacción. El principal problema en este tipo de escenarios, es que no hay experiencia suficiente, ni existen metodologías necesarias para llevar a cabo las pruebas que verifiquen que dichas interfaces carecen de cualquier tipo de deficiencia.

Partiendo de dicha motivación, se ha realizado una metodología que especifica el proceso necesario para la creación de planes y casos de prueba satisfactorios en aplicaciones gráficas de última generación. Dicha metodología está basada en casos de uso, y se descompone en tres fases fundamentales las cuales son: estudio de viabilidad, extracción de conocimiento y generación de documentación y el plan de aseguramiento de la calidad.

Las recientes y novedosas aplicaciones WPF de Microsoft, basadas en la plataforma .NET, son uno de los tipos de aplicaciones sobre las que hay que prestar especial atención a las pruebas de interfaz. Por ello, se ha desarrollado una aplicación basada en esta tecnología con objeto de crear pruebas y desplegarlas sobre la misma, basándose en la metodología definida.

Abstract

The complexity of the computer applications becomes increasingly higher over time with the arrival of new and more powerful technologies. This approach implies having to provide mechanisms for communication with the user, useful and consistent with business logic, as well as intuitive so that through them the user can carry out all the complex functionalities.

As a result of the above, the current focus on creating applications is to provide them very powerful and extremely complex user interfaces, with graphics in two and three dimensions and even with multimedia content. The main problem in such scenarios is that there is not enough experience, and there are no methods to conduct tests and verify that these interfaces lack any sort of deficiency.

Based on that reasoning, a methodology has been made which specifies the process for creating successful plan and test cases about next-generation applications. This methodology is based on use cases, and is broken down into three main phases which are: feasibility study, knowledge extraction and generation of documentation and quality assurance plan.

Recent WPF applications from Microsoft, based on the platform .NET are one of the types of applications that need special attention about interface testing. So we have developed an application based on this technology to create and deploy tests based on the methodology defined.

Índice de Contenidos

Índice de tablas

Índice de Plantillas

Índice de ilustraciones

I

Sobre el documento

I. Sobre el documento

1. Introducción y Motivación

El actual documento, denominado *Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF* ha sido desarrollado como proyecto de fin de carrera.

El planteamiento inicial es el de desarrollar una metodología para el aseguramiento de la calidad sobre aplicaciones de nueva generación (.NET WPF), centrándose concretamente en la parte de interfaz de este tipo de aplicaciones. El marco de actuación de dicha metodología se sitúa dentro de la fase de pruebas del desarrollo de cualquier aplicación de de este tipo.

Por lo anteriormente mencionado, y con motivo de llevar a cabo una puesta en marcha de la metodología definida, el siguiente paso es el de desarrollar una aplicación acorde a las características mencionadas. Dicha aplicación será completada a través de diferentes fases en el desarrollo: análisis, diseño e implementación. Durante las fases mencionadas no se dará la espalda al tema de la calidad, haciendo diferentes pruebas de caja blanca en la fase de implementación.

Finalmente, y una vez terminada la fase de implementación, se pasa a la fase de pruebas. En esta fase se hará un mayor énfasis en el tema de la calidad, llevando a cabo un proceso formal para probar la aplicación, en el cual se hará uso de la metodología definida en un primer momento.

2. Definiciones y acrónimos

A lo largo del documento que se presenta se usan una serie de términos, o diversos elementos técnicos que pueden no contar con el conocimiento de la persona que lea el mismo; de ahí, la necesidad de aclararlos:

2.1. Aclaraciones y Definiciones

- **.NET:** es una arquitectura desarrollada por Microsoft que pretende conseguir la conectividad absoluta en Internet.
- **Acceso permitido a la interfaz mediante programación:** significa que se puede interactuar con una interfaz a través de scripts o programas destinados a tales fines.
- **Administrador de contenidos:** aplicación que permite almacenar distintos tipos de información para posteriormente utilizarla en aplicaciones Web normalmente.
- **Aplicación/Sistema:** una aplicación es un programa informático diseñado para facilitar al usuario la realización de un determinado tipo de trabajo. (Durante el documento se hablará indistintamente de aplicación y sistema).
- **Aplicación visual:** es un tipo de aplicación cuya interfaz con el usuario carga algún tipo de gráfico, ya sea en dos o tres dimensiones.
- **Automatizar:** ejecución automática de tareas, en este caso pruebas de software, haciendo más ágil y efectivo el trabajo y ayudando al ser humano.
- **Bug:** se trata de un error o deficiencia en las aplicaciones informáticas.
- **C# .NET:** se trata de un lenguaje de programación orientado a Web.
- **Caso de uso:** en ingeniería del software, un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

- **Cliente:** destinatario de un producto provisto por el proveedor.
- **Control:** se refiere a los elementos básicos de una interfaz de usuario, a través de los cuales el usuario llevará a cabo la tarea que desee. Por ejemplo un botón.
- **Elementos/Controles identificables unívocamente:** en ocasiones es necesario hacer referencia a algún control, por lo que se espera que todos ellos tengan una propiedad cuyo valor sea diferente de los demás. En caso de que haya más de un control con el mismo valor en dicha propiedad, no se podrá saber a cual de ellos se pretende referenciar.
- **Gestión de configuración:** conjunto de procesos destinados a asegurar la validez de todo producto obtenido durante cualquiera de las etapas de un desarrollo. Estos procesos pueden ser *control de cambios* y *versiones*, lugar de almacenamiento de los productos en cuestión, etc.
- **HTML:** lenguaje de marcado que tiene su base en el lenguaje SGML, utilizado para la creación de páginas Web.
- **Login:** identificador de un usuario dentro de un sistema. Para entrar en un sistema protegido se suele requerir además una contraseña de acceso.
- **Modelo:** es una representación simplificada de la realidad que contiene las características generales de algo que se va a realizar
- **MVC:** patrón de diseño basado en la existencia de un modelo (gestión de datos), una vista (presentación) y un controlador (acciones).
- **Password:** palabra clave necesaria para entrar en un sistema protegido. Esta suele ir acompañada de un identificador de usuario.
- **Prueba:** en el mundo de la ingeniería del software, una prueba es una sucesión de acciones que determina el correcto funcionamiento de aplicaciones informáticas.
- **Requisitos:** se tratan de todos los aspectos y necesidades que debe cubrir un proyecto.
- **Script:** se trata de un guión o conjunto de instrucciones en un determinado lenguaje. Permiten la automatización de pruebas entre otras muchas cosas.

- **Usuario:** un usuario es la persona que utiliza o trabaja con algún objeto o que es destinataria de algún servicio público o privado, empresarial o profesional.
- **VB .NET:** se trata de un lenguaje de programación orientado a Web.
- **XAML:** es un nuevo y potente lenguaje de programación para diseñar interfaces de usuario. Dicho lenguaje está basado en XML.
- **WPF:** es una nueva tecnología de Microsoft para el desarrollo de aplicaciones de nueva generación.

2.2. Acrónimos

Acrónimo	Descripción
C#	C Sharp
CU	Caso de Uso
CUOA	Caso de Uso de Operaciones de Acceso
CUOI	Caso de Uso de Operaciones e Interacción con el sistema.
ECGD	Extracción de Conocimiento y Generación de Documentación Preliminar
EV	Estudio de Viabilidad
HTML	HyperText Markup Language
JEC	Jefe Encargado de la Calidad
MVC	Modelo Vista Controlador
PAC	Plan de Aseguramiento de Calidad
Q.A	Quality Assurance
SW	Software
UI	User Interface
UR	Requisito de Usuario
URC	Requisitos de Usuario de Capacidad
URR	Requisitos de Usuario de Restricción
VB	Visual Basic
WPF **XAML**	Windows Presentation Foundation eXtensible Application Markup Language

II

Estado del Arte

II. Estado del arte

1. Introducción y objetivos

Históricamente el tema de la calidad no ha tenido demasiado protagonismo en la industria del software. Esta tarea se consideraba innecesaria, puesto que en la misma había un gasto económico excesivamente elevado.

Esta visión ha ido cambiando con el paso de los años, aumentando el esfuerzo y los gastos en términos de aseguramiento de la calidad. Este cambio de mentalidad no ha surgido por azar, sino que, como consecuencia de la experiencia acumulada durante años se ha visto que es posible obtener mejores resultados utilizando técnicas de calidad, incluso asumiendo el coste que supone.

Este proyecto de final de carrera estará estrechamente relacionado con el tema de la calidad, tratando de solucionar carencias en este sentido en los desarrollos software. Las diferentes temáticas contenidas en el actual apartado de *estado del arte* serán relativamente variadas en cuanto a su tipología, pero estrecha e inevitablemente ligadas en el marco empresarial.

A continuación se exponen los puntos más importantes a tratar en el actual apartado:

- Calidad de software
- Aplicaciones WPF
- Metodologías

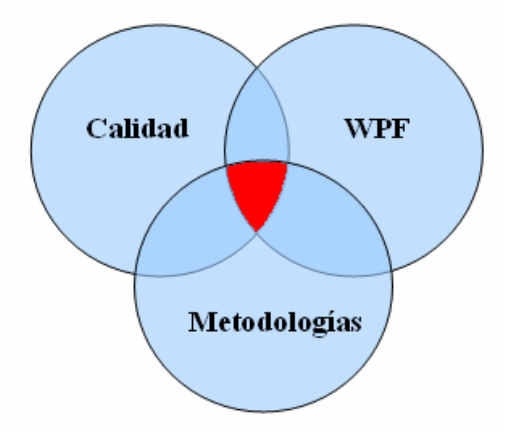


Ilustración 1: Fusión de contenidos

Como se puede observar en la anterior ilustración, las tres tipologías de contenido se pueden solapar de tal forma que entren en un marco común. En el caso en el que nos encontramos, se tratará de llevar a cabo un aseguramiento de calidad sobre las interfaces de usuario de aplicaciones WPF, aplicando alguna metodología apropiada para ello.

2. Calidad de Software

Todo desarrollo de software tiene como objetivo la satisfacción del cliente para el que haya sido elaborado. Para conseguir que dicho cliente esté satisfecho se deberán no sólo aplicar prácticas y metodologías en términos de desarrollo, sino también en términos de calidad. Cabe mencionar que la calidad de un producto como el software no se puede improvisar ni añadir en el último momento. Esto es así ya que dicho proceso de calidad deberá comenzar desde que se marca el hito de comienzo del proyecto y finaliza cuando el cliente da su conformidad de lo obtenido, pudiendo incluso alargarse más tiempo después de la entrega (Sierra, M., 2008).

Hay numerosas definiciones de calidad por el amplio significado y contexto de la palabra. Una posible definición de calidad en el contexto de desarrollo de software

sería el nivel de satisfacción que presenta un producto o un servicio con respecto a una serie de criterios establecidos.

2.1. Necesidad de procesos de calidad en el desarrollo de SW

Normalmente la complejidad de un desarrollo software es bastante elevada. Esto conlleva que puedan existir numerosos errores de diferente tipología. A medida que se va desarrollando una aplicación sin comprobar si se está haciendo de forma adecuada, se pueden ir acumulando errores que a posteriori serán muy difíciles de solucionar (Sierra, M., 2008).

Hay que tener en cuenta que además una única persona no suele ser la encargada de realizar todo el desarrollo sino que suele haber varias encargadas, por lo que será incluso más complicado que la integración de las partes esté libre de errores.

En caso de no seguir ningún proceso de calidad, podrían ocurrir dos cosas, entre otras tantas:

- El propio equipo de desarrollo se da cuenta de que algo no funciona como debería, teniendo que estudiar qué cosas habría que modificar, y qué dependencias hay entre ellas, teniendo que volver a planificar. En muchos casos esto puede ocurrir en una fase muy avanzada del proyecto por lo que se puede hacer inabordable y suele implicar no cumplir con los plazos establecidos. Aún así, la entrega de dicho producto en caso de producirse, puede seguir conteniendo errores de funcionalidad o de cualquier otro tipo ya que no se ha seguido ningún proceso que garantice la calidad de la misma.

- El cliente es el que experimenta un comportamiento no deseado por parte de la aplicación. En estos casos el cliente podrá rechazar el proyecto o exigir una modificación del mismo a posteriori. El hecho de tener que modificar la aplicación a posteriori, supondría un esfuerzo elevado y en

muchos casos seguirían surgiendo nuevos fallos en la aplicación, haciendo que el cliente perdiera la confianza que tiene depositada en la empresa desarrolladora.

Para evitar en mayor medida todo lo mencionado anteriormente, será necesario seguir procesos de calidad. Un proceso de calidad es un conjunto de actividades enlazadas entre sí durante el ciclo de vida del proyecto, con el objetivo de reducir al máximo los posibles errores de software, o comportamientos indeseados.

2.2. Disminución de Costes en los desarrollos revisados por calidad

Como se ha mencionado anteriormente, el hecho de hacer un trabajo bien a la primera puede disminuir los costes globales de desarrollo de una aplicación. A continuación se muestra algunos factores determinantes (Kaner, Falk & Quoc, 1993):

- Los errores se descubren más temprano, por lo que serán más fácil de solucionar.
  - Menos horas de personal trabajando para solucionar errores de menos complejidad.
  - Mayor probabilidad de cumplimiento de plazos debido a la constante revisión por calidad.

- Personal cada vez más cualificado porque está acostumbrado a seguir procesos de calidad.
  - La calidad de futuros desarrollos puede verse incrementada.

- Satisfacción por parte de los clientes. Cuanta mayor calidad haya en los desarrollos software, se dotará de mayor reconocimiento y prestigio a la empresa. Ese factor diferenciador puede ser clave y servir como eslogan de competitividad empresarial. Este hecho puede reportar grandes beneficios en forma de:
  - Más trabajos para antiguos clientes.

- Nuevos clientes.
- Ampliación de la cuota de mercado.

2.3. Desarrollo de software en empresas

En los negocios, el desarrollo de software normalmente es llevado a cabo por un grupo de personas trabajando conjuntamente. Cada una de dichas personas deberá poseer determinadas características para poder desempeñar satisfactoriamente el papel o rol que se le ha asignado en el desarrollo del proyecto.

Para que un desarrollo software se lleve a cabo de forma satisfactoria deberá previamente haber sido planificado. Por ello se deberán establecer diferentes fases de desarrollo en cada una de las cuales se vaya completando el mismo. En la planificación del proyecto, además se deberán asignar recursos de todo tipo por cada fase (humanos, materiales, etc.), que harán que se disponga de los medios necesarios para llevarla a cabo.

Cumplir con dichas fases previamente planificadas será primordial para conseguir todos los objetivos planteados desde un principio.

2.3.1. Responsabilidades de calidad por cada rol

A continuación se expondrán los roles más comunes en este tipo de escenarios. En la práctica, no suele haber una correspondencia 1-1 entre persona y rol, sino que en la mayoría de las compañías mismas personas suelen desempeñar más de un rol:

Por cada rol, se expondrán una serie de responsabilidades asignadas que estarán directa o indirectamente relacionadas con la calidad del proyecto (Villarroel, R. H., 2007).

2.3.1.1. Administrador de proyecto

El administrador de proyecto es la persona que administra y controla los recursos asignados a un proyecto, con el propósito de que se cumplan correctamente los planes definidos. Por recursos se refiere a recursos humanos, económicos, tecnológicos, espacio físico, etc.

Algunos de los objetivos de un administrador de proyecto son los siguientes:

- Tener el producto dentro de los plazos marcados, con el menor gasto posible y con la mayor calidad.
- Terminar el proyecto con los recursos asignados.
- Coordinar los esfuerzos generales del proyecto, ayudando a cada uno de sus integrantes a cumplir sus objetivos particulares. Al final, se cumplirá el objetivo general.
- Cumplir con éxito las diferentes fases de un proyecto, utilizando herramientas de administración.
- Cumplir con las expectativas del cliente, lo cual será el fin propuesto desde un principio.

2.3.1.2. Analista

La palabra “análisis” se refiere a una característica típicamente relacionada con la inteligencia humana. Esta se refiere a la habilidad de poder estudiar un problema de una complejidad determinada, descomponiendo el mismo en subproblemas de menor complejidad. Como el experto en el problema es el cliente, se hace necesario trabajar junto a él para realizar la especificación correctamente.

Un analista es una persona que entiende qué quieren los clientes y cómo especificarlo en términos que un programador o diseñador puedan entender. Por ello, la mayor responsabilidad del analista será extraer correctamente el conocimiento del cliente, y especificar formalmente el mismo. Como consecuencia de un insuficiente

esfuerzo dedicado a conocer y especificar el sistema que desea el cliente, los desarrolladores construirán sistemas que no cuentan con las características que el cliente desea.

2.3.1.3. Diseñador

Es el encargado de crear una estructura interna del sistema, denominada arquitectura, y de la definición de relaciones entre subsistemas. Para ello deberá basarse en los requisitos creados en la fase de análisis.

El diseño obtenido en esta fase deberá tener el suficiente nivel de detalle para que en un futuro los programadores no tengan que tomar ningún tipo de decisión de implementación. Esto evita numerosos errores ya que los programadores simplemente tendrán que implementar el sistema definido en dicho modelo.

En ocasiones no es suficiente con realizar un modelo exhaustivo de lo que será la futura aplicación, sino que además se debe proporcionar el máximo detalle en diversos aspectos, como por ejemplo:

- Seleccionar el lenguaje y paradigma apropiado con el que será implementada la aplicación.
- Seleccionar el método de almacenamiento apropiado para las estructuras de datos, por ejemplo, estructuras de datos vs. sistemas de archivos vs. SABD.

Por todo lo mencionado, la responsabilidad del diseñador está muy clara, y es la de llevar a cabo un diseño global del sistema, especificando y diseñando todas y cada una de las partes, excluyendo la ambigüedad. De esta forma los propios desarrolladores no tendrán que improvisar en la fase de implementación.

Ocasionalmente podría escribir diferentes planes de pruebas de alto nivel, ya que el diseñador tiene conocimientos globales de la aplicación y puede saber a priori qué cosas sería conveniente verificar.

2.3.1.4. Programador

Los programadores deben convertir la especificación del sistema en código fuente ejecutable. Para ello deberán respetar las decisiones tomadas y el modelo llevado a cabo por los diseñadores, no debiendo tomar decisiones propias al respecto.

Uno de los principales objetivos de los programadores durante su trabajo debe ser la de reducir la complejidad del software. Algunos de los beneficios que implican la reducción de la complejidad del programa son:

- Menor cantidad de problemas de testeo.
- Aumento de la productividad de los programadores.
- Aumento de la eficiencia en la manutención del programa.
- Aumento de la eficiencia en la modificación del programa.

2.3.1.5. Asegurador de calidad

En la actualidad, los factores dominantes en la administración de proyectos de software son los tiempos y costos de desarrollo, existiendo buenas razones para ello. Los tiempos y costos de desarrollo son con frecuencia, muy grandes. Por ello, la administración se suele concentrar en tratar de resolver dichos problemas. Sin embargo, existe un gran peligro en esto. En la medida que crece la presión por cumplir con las fechas estipuladas, y reducir los costos, es la calidad del producto la que sufre, yendo en decremento.

Por lo mencionado anteriormente, el asegurador de calidad es aquel rol responsable de que la calidad del producto no se vea perjudicada por otros factores como lo pueden ser el tiempo y coste, y cumpla con las especificaciones del cliente.

2.3.1.6. Analista de pruebas

El desarrollo de un sistema de software requiere la realización de una serie de actividades de producción. En dichas actividades existe la posibilidad de que aparezcan errores humanos. Por ello, el desarrollo de software considera una actividad que apoye el proceso de detección y eliminación de los errores y defectos del sistema en construcción.

El objetivo principal de la labor del analista de pruebas es el de diseñar pruebas que de forma sistemática, permitan detectar para posteriormente eliminar diferentes clases de errores, realizando esto con la mínima cantidad de tiempo y esfuerzo. Los objetivos específicos en la labor del mismo son los siguientes:

- Aplicar métodos para diseñar casos de prueba efectivos.
- Construir buenos casos de prueba que tengan altas probabilidades de encontrar errores aún no descubiertos.
- Demostrar que las funciones del sistema parecen estar funcionando de acuerdo a sus especificaciones.
- Proveer una buena indicación de la confiabilidad del software y algunas indicaciones de la calidad del software.

2.3.1.7. Ingeniero de validación y verificación

El objetivo principal del proceso de validación y verificación de software es el de analizar y probar de forma completa el software durante el desarrollo para determinar si su funcionalidad es llevada a cabo correctamente.

Dependiendo de las actividades de validación y verificación llevadas a cabo respecto a la funcionalidad y rendimiento del software, es posible identificar algunos objetivos:

- Correctitud: en qué grado el producto está libre de errores.
- Necesidad: en qué grado lo que hay en el producto es necesario.

- Suficiencia: en qué grado el producto es completo.
- Rendimiento: en qué grado el producto satisface los requisitos de rendimiento.

2.3.1.8. Documentador

Durante el proceso de desarrollo de software, se genera una gran cantidad de documentación. La documentación sirve, entre otras cosas, para conocer la historia del proyecto.

El objetivo principal del documentador es generar elementos (documentación) que actúen como medio de comunicación entre los miembros del equipo, incluyendo el cliente. Este rol será responsable de la completitud y claridad de los mismos generados, pues es parte fundamental en la comunicación entre los diferentes miembros del equipo.

2.3.2. Tareas de calidad por etapa en un desarrollo software

Para la obtención de un producto software, como se ha mencionado anteriormente, hay que ir desarrollándolo a través de una serie de fases. Las principales fases que suelen repetirse en gran cantidad de proyectos de este tipo son usualmente las que se enumeran a continuación:

- Análisis.
- Diseño.
- Implementación.
- Pruebas.
- Mantenimiento y mejoras.

Como se puede observar hay una fase destinada a realizar pruebas sobre el producto obtenido. A continuación veremos cómo esto no es exactamente así, puesto

que generalmente las pruebas tienen un ámbito mucho más global, y están enmarcadas en más fases del propio proyecto.

Como se ha mencionado anteriormente, aunque se puedan llevar a cabo pruebas en numerosas fases del ciclo de vida del proyecto, el coste de encontrar errores y solucionarlos incrementa dramáticamente según progresa el proyecto (Sierra, M., 2009).

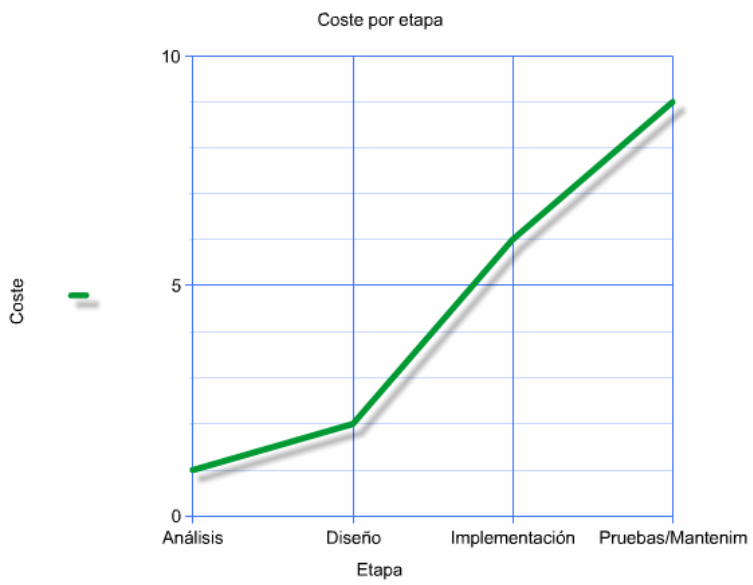


Ilustración 2: Coste de solucionar defectos en fases Sw.

A continuación se expone el cometido fundamental de cada una de las fases mencionadas, exponiendo además las tareas generales de calidad que se suelen realizar en cada una de ellas (Kaner, Falk y Quoc, 1993):

2.3.2.1. Fase de Análisis

Obtención de una especificación detallada del sistema de información que satisfaga las necesidades de los usuarios, lo cual se tomará como base del posterior diseño.

Para ello será necesario recoger los requisitos que se deben cumplir, dejando a un lado detalles de diseño e implementación a los desarrolladores.

Las **pruebas** en esta fase de análisis se realizarán sobre los propios requisitos. Los revisores deberían evaluar el documento de requisitos para verificar que sean correctos. Un requisito se considera correcto cuando refleja lo que se espera de un sistema. Además hay otra serie de enfoques a tener en cuenta, en función a los cuales un requisito debería o no ser incluido:

- **Completitud:** comprobar que no se espera nada más del sistema, así como verificar que todos los requisitos especificados son necesarios.

- **Consistencia:** comprobar que los requisitos no sean lógicamente incompatibles o contradictorios entre sí. Un conjunto de requisitos contradictorio no es implementable.

- **Factibilidad:** comprobar si los requisitos pueden ser llevados a cabo sobre el sistema en cuestión. Un requisito puede no ser factible por temas de velocidad, asumiendo que el hardware trabaja más rápido de lo que realmente lo hace, por temas de memoria, tiempo, etc.

- **Requisitos razonables:** debe haber un equilibrio en cuanto a determinados parámetros tales como velocidad de desarrollo, coste de desarrollo, rendimiento, fiabilidad y uso de memoria. Cumplir con cada uno de ellos por separado podría ser factible, pero si los requisitos son muy restrictivos, refiriéndose a todos los parámetros mencionados en su conjunto, quizás nunca se puedan cumplir de forma global.

- **Verificables:** un requisito es verificable si existe un proceso finito y no costoso para demostrar que el sistema cumple con el requisito. Un requisito ambiguo no es, en general, verificable.

2.3.2.2. Fase de Diseño

El diseño consiste en producir un modelo o representación técnica del software que se va a desarrollar. Éste depende del documento de requisitos que lista las capacidades que debe tener el producto, obtenido en la etapa anterior. Si no existe dicho documento o está incompleto, los diseñadores tendrán que tomar sus propias decisiones acerca de las capacidades del producto.

Acorde al modelo tradicional de desarrollo de software, la codificación no debe comenzar hasta que el diseño se haya completado. Por ello, al no haber escrito código las **pruebas** en esta fase de diseño se llevarán a cabo sobre ideas. Los revisores deberán explorar las siguientes cuestiones:

- **Calidad en el diseño:** el diseño desarrollado deberá permitir la obtención de un producto mantenible, compacto y eficiente. Además de deberán poder realizar pruebas para verificar si cumple con determinados requisitos de calidad.

- **Completitud:** un diseño completo es aquél que especifica todas las relaciones entre módulos, cómo se pasan los datos, qué ocurre en circunstancias especiales o qué estado inicial hay que asumir para cada módulo, entre otras cosas. Es decir, definir todo lo necesario para no dotar de ningún grado de libertad de decisión a los programadores en la fase de implementación.

- **Factibilidad:** es posible la obtención del producto en cuestión tomando como base el diseño realizado.

El pseudocódigo es un lenguaje artificial que combina el lenguaje de codificación con un lenguaje natural. Normalmente los diseñadores encuentran conveniente describir el diseño en este lenguaje casi tan formal como un lenguaje de programación, con muchas de las mismas estructuras. Por ello, se pueden ayudar en la tarea de verificar la calidad del diseño, de ciertas herramientas existentes como por

ejemplo programas que buscan deficiencias en el mismo. A estos programas se les suele denominar *analizadores de pseudocódigo*.

2.3.2.3. Fase de Implementación

Durante esta fase se codifican, en el lenguaje de programación escogido, cada una de las especificaciones obtenidas en la fase de diseño de la aplicación. En esta fase es donde se obtiene una representación física del producto, la cual está aún muy lejos de ser la representación definitiva.

No sólo es suficiente con codificar, sino que dentro de la propia fase de implementación también tiene lugar la creación y ejecución de **pruebas** que garantizan que lo que se ha codificado funciona correctamente.

Las pruebas mencionadas que tendrán lugar en dicha fase serán catalogadas en su mayoría como pruebas de *caja blanca*, ya que este es el tipo de pruebas para las que los programadores están bien equipados. Dichas pruebas de caja blanca serán descritas en posteriores apartados.

2.3.2.4. Fase de Pruebas

Una vez que se termina la fase de implementación, el programa se pasa al equipo de pruebas para que éste realice más pruebas sobre el mismo. Durante esta nueva fase de pruebas, se deberán realizar numerosos casos de prueba ya que en muchas ocasiones el hecho de encontrar y solucionar un error puede provocar introducir otros por otro lado.

Una vez que la aplicación en cuestión pase todas las pruebas pertinentes, ésta podrá ser entregada al cliente con total seguridad de que todos los requisitos impuestos por el mismo desde un primer momento han sido completamente satisfechos.

Las pruebas que se realizan durante esta fase de desarrollo suelen ser pruebas de *caja negra*, pues fundamentalmente su cometido será el de verificar la funcionalidad. A continuación se describe una secuencia de eventos que es bastante usual llevar a cabo una vez que las pruebas de caja negra han comenzado:

- Planificación de las pruebas: en esta fase se deberá llevar a cabo un diseño de casos de prueba y planificación de ejecución de los mismos. Se puede planificar tan pronto como se tenga el documento de requisitos, pero normalmente el diseño detallado de las pruebas se suele realizar durante el primer ciclo de las pruebas.

- Pruebas de aceptación: cada vez que se reciba una versión del programa, se deberá comprobar si es lo suficientemente estable para ser probado. Si este tiene comportamientos no deseados "a la menor provocación", no se deberá seguir perdiendo el tiempo con ello.

  Normalmente se debe estandarizar este tipo de pruebas de aceptación para remitírselas a los programadores. Con esto se pretende que ellos mismos lleven a cabo dichas pruebas antes de que se envíe la versión al equipo de pruebas. Este hecho permitirá no perder el tiempo con versiones que aún son poco maduras.

- Evaluación inicial de estabilidad: no se tata de encontrar errores de por sí en este apartado. Se tratará de decidir en qué áreas del programa se confía menos. El motivo de esta evaluación inicial es el de indagar qué partes de la aplicación pueden necesitar más pruebas, más o menos exhaustivas. Si el programa parece débil en un área que es difícil de probar, se esperará tardar bastante tiempo en probarla.

  Examinar los manuales existentes junto con el programa puede ser un buen comienzo. En función a los datos obtenidos, puede ser oportuno llevar a cabo unas pocas pruebas que se espera que provoquen que el

programa falle. Al final de la evaluación inicial, se deberá tener una idea global acerca del programa, en términos de la exhaustividad con la que se debe probar el mismo y estimando los errores que puede contener. En ocasiones estas estimaciones pueden formalizarse numéricamente pudiendo obtener datos del tipo personas-horas trabajadas u horas-área.

- Pruebas funcionales: se trata del conjunto de pruebas que tienen lugar sobre el sistema para verificar que el mismo cumple con todos los requisitos planteados, habiendo sido codificado correctamente.

  Esta serie de pruebas, normalmente catalogadas como de caja negra, será desarrollada en posteriores apartados.

- Pruebas beta: cuando un programa y su documentación parecen estables, es momento de obtener información del usuario. Las pruebas beta están destinadas hacia la gente que representa el mercado en el que nos encontramos.

  La idea fundamental es que los usuarios utilicen el producto de la misma forma que lo harían si hubieran comprado la versión final del producto. Con ello lo que se pretende es obtener su opinión y comentarios acerca de la aplicación, además de encontrar posibles deficiencias que no hayan sido detectadas hasta el momento.

  Aún no se espera que el producto esté libre de errores, por lo que hay que dar tiempo suficiente a esta fase ya que hasta que los usuarios no tengan destreza en su utilización, no profundizarán lo suficiente como para poder reportar aspectos a tener en cuenta, posibles deficiencias, etc.

  Los usuarios que disfrutan de una versión beta de una aplicación, suelen estar inducidos a ello por algunos de los siguientes aspectos, entre otros tantos:

- Es el único producto existente de ese tipo.
- Reciben retribuciones económicas de algún modo. En ocasiones, se permite dejar disfrutar gratis de la aplicación o pagarla a precios reducidos si se han aportado cosas importantes.
- Soporte técnico gratuito.

- Pruebas de liberación: en las pruebas de liberación se deben recolectar todas las cosas que se le van a entregar al cliente, comprobando que son correctas. Una vez hecho esto se copiarán, archivarán y se liberarán.

  Las pruebas más comunes de liberación son las de comprobación de que los entregables han sido correctamente copiados en el tipo de soporte electrónico que proceda. Para ello se suelen hacer comparaciones binarias entre los ficheros copiados con los de la versión que se declaró como buena durante la ronda final de pruebas.

  Es altamente recomendable incluir en este tipo de pruebas comprobaciones para que los ficheros estén a salvo de virus. En caso de que el entregable esté comprimido, se deberá comprobar si su contenido es correcto, posteriormente se deberá instalar la aplicación y ejecutarla para finalmente reiniciar. Una vez hecho esto se deberá comprobar si después de la instalación hay algún tipo de virus en la máquina.

- Pruebas de aceptación final y certificación: si el producto ha sido desarrollado por contrato, el cliente deberá ejecutar unas pruebas de aceptación cuando este le sea liberado. Hay que estar seguro de que el programa pasa las pruebas antes de entregárselo al cliente.

  En ocasiones se requiere certificación en lugar de aceptación. La certificación tiene que ser llevada a cabo por terceras entidades, tales como agencias de certificación. El objetivo que se persigue con las

pruebas de certificación es que cierta entidad garantice que se cumplen las especificaciones que se enunciaron en el contrato. Por ello, el nivel de exhaustividad de las pruebas vendrá dado por lo especificado por contrato.

2.3.2.5. Fase de mantenimiento

Una gran cantidad de dinero destinada al proyecto se invertirá durante esta fase. Este hecho tiene lugar ya que en la mayoría de los casos, aunque ya se haya liberado el producto, éste debe seguir siendo modificando con objeto de añadir funciones para mejorar el mismo, diferenciarse del resto de productos del mercado o simplemente para resolver incidencias.

En esta fase, las pruebas realizadas sobre el software podrán ser de cualquiera de los tipos anteriormente mencionados.

Según el libro publicado por Martin & McClure:

- La fase de mantenimiento produce aproximadamente un 60% de los gastos totales del proyecto.
- El 20% del presupuesto del mantenimiento se utiliza en solucionar errores.
- El 25% se emplea adaptando el programa para que éste funcione con nuevo hardware o software.
- El 8% se emplea modificando la documentación.
- El 5% se emplea en mejoras de rendimiento.
- El 42% se emplea en realizar cambios requeridos por los usuarios.

2.4. Pruebas en ingeniería del software

Hay muchas definiciones de pruebas en ingeniería del software, sin embargo todas ellas se reducen esencialmente a una misma cosa: ejecutar un determinado software de forma controlada con objetivo de evaluar un programa o sistema, y determinar si cumple con las pretensiones establecidas por el cliente.

Hay varias técnicas para llevar a cabo las pruebas sobre un determinado sistema. Cada una de ellas será más importante en unas u otras fases del desarrollo del producto:

- Pruebas de caja blanca (White Box): estas pruebas se basan en el conocimiento que se tiene acerca de la implementación del sistema.

- Pruebas de caja negra (Black Box): estas pruebas se basan en las observaciones que se pueden realizar a través de la interfaz de la aplicación y su comportamiento, manteniendo al margen cómo se ha implementado la misma.

- Pruebas de caja gris (Gray Box): estas pruebas son un híbrido entre pruebas de caja blanca y pruebas de caja negra. En ellas se suele utilizar ciertos conocimientos acerca de cómo ha sido implementado un determinado elemento de la aplicación para guiar las pruebas basadas en la observación de la interfaz.

2.4.1. Objetivos y límites de las pruebas de calidad

En los planes de pruebas realistas se debe seleccionar un pequeño subconjunto de casos de prueba de un enorme conjunto de posibilidades. No importa lo duro que se trabaje y lo buenos y representativos que sean nuestros casos de prueba porque siempre se dejará fuera una gran cantidad de pruebas importantes. Por ello, hay que ser consciente de que nunca se encontrará el último error en un programa (Kaner, Falk y Quoc, 1993).

Por ello, se puede decir que el objetivo fundamental de probar es encontrar errores, manteniendo al margen el obtener una aplicación carente de ellos ya que nunca sabremos cuando se ha llegado a este punto. Buenos casos de prueba son más susceptibles de encontrar errores, o más si cabe de encontrar errores críticos.

Unas de las principales razones por las que es imposible hacer pruebas completas sobre una aplicación son:

- El dominio de las posibles entradas es demasiado grande para ser probado.
- Hay demasiados posibles caminos a través de la lógica de un programa para ser probados.
- Las interfaces de usuario son demasiado complejas para poder probar todo completamente.

2.4.2. Tipos de pruebas

A continuación se expondrán una serie de pruebas, las cuales serán de diferente tipología, y serán clasificadas en función a si son de *caja negra* o de *caja blanca* (Kaner, Falk y Quoc, 1993):

2.4.2.1. Pruebas de caja negra

Algunas de las pruebas de caja negra más importantes y utilizadas son:

- **Pruebas de verificación de las especificaciones:** se comprueba que el comportamiento del programa es el que debe ser.

- **Pruebas de usabilidad:** se estudia qué cosas harán que el programa sea más fácil y cómodo de utilizar por parte de los clientes más representativos.

- **Pruebas de rendimiento:** se identifican tareas y se mide cuánto tiempo tardan en ejecutarse.

- **Pruebas de carga:** este tipo de pruebas estudian el comportamiento de la aplicación cuando esta trabaja al límite:
  - Pruebas de volumen: se alimenta al programa con gran cantidad de datos.
  - Estrés: se comprueba la respuesta del sistema ante picos o ráfagas de actividad.

- **Pruebas de almacenamiento:** se verifica cómo administra el sistema la memoria.

- **Pruebas de concurrencia:** se suelen realizar pruebas multi-hilo para comprobar si el sistema está capacitado para ello.

- **Pruebas de seguridad:** se comprueba la facilidad con la que un usuario desautorizado obtiene acceso a la aplicación.

2.4.2.2. Pruebas de caja blanca

Algunas de las pruebas de caja blanca más importantes y utilizadas son:

- **Pruebas unitarias:** el programador puede probar el programa por partes, escribiendo código de prueba especial que proporcione valores de entrada interesantes a diferentes módulos aislados de la aplicación.

- **Pruebas de cobertura:** con este tipo de pruebas el programador podrá encontrar qué partes del programa son ejecutadas por las pruebas. Se pueden encontrar qué líneas de código, qué saltos, o qué caminos no han sido probados aún, pudiendo a su vez añadir pruebas en las que se sepa que cierta parte de código aún no ha sido ejecutada, forzando a que estas sean probadas.

- **Control de flujo:** el programador sabe qué se supone que un programa hará después, como consecuencia de su estado actual. Él podría modificar el programa para que éste siempre informe qué va haciendo o podría utilizar un depurador. Con esto sabrá el orden en que se han ejecutado ciertas líneas de código, obteniendo otra serie de datos como valores de variables en cada momento, posibilidad de cambiar dichos valores, etc.

- **Integridad de los datos:** se trata de pruebas en las que se hacen comprobaciones de si los datos manejados por la aplicación son modificados por los módulos adecuados.

- **Pruebas sobre algoritmos específicos:** si un programa utiliza algoritmos conocidos para realizar cálculos complicados, posiblemente ya existan pruebas de calidad sobre ese tipo de algoritmos.

2.4.3. Automatización de casos de prueba

En ocasiones puede ser conveniente automatizar determinados casos de prueba. Un falso tópico bastante extendido es el que menciona que la automatización de pruebas disminuirá los costes asociados a las pruebas (Fewster y Graham, 1999).

- Pruebas manuales: son aquellas pruebas que típicamente funcionan mejor (o no se pueden realizar de otra forma) si son realizadas directamente por personas.

- Pruebas automáticas: son aquellas pruebas que pueden ser programadas y ejecutadas sin la intervención de una persona. Este tipo de pruebas son típicamente de naturaleza repetitiva, pudiendo ser ejecutadas con frecuencia y regularidad, dando constante información a los desarrolladores acerca de qué cosas funcionan y cuáles no.

La decisión de automatizar pruebas no se suele realizar con objetivo de disminuir costes, sino más bien se miran otros factores, de los cuales se pueden obtener los siguientes beneficios, entre otros muchos (Dustin, Rashka y Paul, 1999):

1. **Mejor organización de las pruebas:** para llevar a cabo la automatización de pruebas, se deberán analizar en más detalle. Al crear casos de pruebas deben formularse preguntas del tipo: ¿es de carácter repetitivo?, ¿necesito ejecutarla cada cierto tiempo?, ¿tiene alguna semejanza esta prueba a pruebas existentes?, ¿de qué forma la podría automatizar?

2. **Encontrar un mayor número de defectos en menos tiempo:** posibilidad de encontrar defectos en el software que tal vez no hubieran sido encontrados utilizando sólo pruebas manuales, debido a limitantes de tiempo.

3. **Habilitación de pruebas de regresión:** existiendo determinadas pruebas, y habilitados por una herramienta de automatización, será posible probar todas ellas de nuevo por completo cada que se genere un nuevo build del sistema en cuestión. Esto es de vital importancia, ya que tras correcciones de defectos se pueden incorporar nuevos.

4. **Mayor confiabilidad en los resultados:** el sistema de automatización está al margen de situaciones diversas que puedan ocurrir. Este no se cansa, nunca tiene prisa, y mientras las pruebas o su información no cambie, deben de obtener siempre el mismo resultado; son consistentes, confiables, y repetibles. Como seres humanos los anteriores aspectos mencionados nos pueden influir, afectando a la capacidad de ser eficiente en pruebas rutinarias. La automatización de pruebas repetitivas que requieren una ejecución frecuente, permite tiempo para integrar pruebas más complejas, probar nuevas funciones dentro de la aplicación y su integración con el resto del sistema.

5. **Capacidad para aplicar pruebas complicadas:** algunos tipos de prueba son difíciles de aplicar o muy complicadas de ejecutar de manera manual; entre este tipo de pruebas podemos encontrar aquellas en las que es necesario el acceso a la base de datos para verificar que la información del sistema sea correcta, o tal vez sea preciso hacer cálculos manuales para verificar la validez de los resultados arrojados por el sistema. Muchas herramientas de automatización proporcionan estas funcionalidades. Además, los sistemas de automatización pueden ayudar a introducir grandes cantidades de información, configurar la versión de

prueba de la base de datos, y generar información aleatoria entre otras cosas.

3. Tecnología WPF

Con la llegada de la versión 3.0 (actualmente versión 3.5) del Framework de .Net, no sólo aparecen nuevas tecnologías como Windows Presentation Foundation (WPF) sino que además aparece un nuevo lenguaje con el objetivo de ayudar en el desarrollo de programas modernos. Se trata de XAML (eXtensible Application Markup Language).

Es un **lenguaje declarativo** que se basa en XML y que está **pensado para desarrollar interfaces en WPF**. XAML permite separar la interfaz de usuario de la parte lógica de la aplicación, utilizando XAML para la interfaz de usuario y C# o VisualBasic.Net para la lógica de los programas (Microsoft Corporation, 2007).

Antes de comenzar a detallar XAML, se debe dar una pequeña explicación de qué es el Framework 3.0 de .Net así como también que es WPF.

3.1. Plataforma .Net

.NET es el nombre que ha dado Microsoft a su forma de ver el futuro de las aplicaciones. Se pretende un desarrollo de software con énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Esta visión se centra en un mundo en el cual las aplicaciones se ejecutan de modo distribuido, a lo largo de todo Internet, y son accesibles desde múltiples dispositivos.

Esta visión no surge al azar, si no que tiene varias motivaciones. La primera motivación, se debe a que al igual que la programación orientada a objetos, la programación distribuida fomenta la reutilización de software. De hecho, lo lleva un paso más adelante, ya que no sólo podemos reutilizar nuestro propio código, o aquellas librerías de las se dispone, sino que se pueden reutilizar recursos disponibles en distintas ubicaciones de Internet.

Por otra parte, el poder acceder a las aplicaciones desde cualquier sitio y desde cualquier dispositivo, es una aspiración de Microsoft, que con la tecnología .NET puede llevarse a cabo de forma sencilla, mediante la utilización de servicios web.

Microsoft divide .NET en tres apartados muy distintos:

- **La infraestructura .Net.**
- **Los servicios de Internet.**
- **Programas .Net**.

Lo más visible de .Net es su infraestructura. Se trata de todas las tecnologías que conforman el nuevo entorno que permite a los desarrolladores crear y ejecutar nuevas aplicaciones.

3.2. .Net Framework

Se llama Framework o entorno de trabajo, a las Bibliotecas de Clase Base, (también llamadas BCL) y al Common Language Runtime, (CLR).

3.2.1. Bibliotecas de clase Base (BCL)

Las bibliotecas de clase son una nueva estructura jerárquica de clases que envuelven diversas funcionalidades y que están disponibles para cualquier lenguaje .Net (Visual C++.Net, Visual Basic.Net, Visual C#.Net, ASP.Net, etc.).

Las funcionalidades que envuelve la BCL son la mayoría de las operaciones básicas que se encuentran involucradas en el desarrollo de aplicaciones, como por ejemplo: administración de memoria, manejo de tipos de datos unificado, operaciones aritméticas, etc.

3.2.2. Common Language Runtime (CLR)

El CLR es el verdadero núcleo del Framework, es el entorno que usan las aplicaciones escritas en diversos lenguajes en tiempo de ejecución. El CLR gestiona la ejecución de cada ejecutable encapsulándolo, separándolo de otros procesos de la máquina. Ofrece una interoperabilidad multi-lenguaje, o lo que es lo mismo, la característica de que cada aplicación escrita en diferentes lenguajes pueda interactuar sin inconvenientes.

Para esto, la herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un código intermedio (MSIL, Microsoft Intermediate Lenguaje). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear ese código MSIL compatible con el CLR.

Para ejecutarse es necesario un segundo paso, un compilador JIT (Just-In-Time) es el que genera el código máquina real que se ejecuta en la plataforma del cliente.

De esta forma, con .Net se consigue independencia de la plataforma hardware. La compilación JIT la realiza el CLR a medida que el programa invoca métodos, el

código ejecutable obtenido se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.

3.2.3. .NET Framework 3.0

Microsoft .Net Framework 3.0 (también conocido como WinFX), fue un nuevo modelo de programación para Windows que combina la potencia del Framework 2.0 con cuatro nuevas tecnologías (Wikipedia, 2008):

- Windows Presentation Foundation (WPF).
- Windows CardSpace.
- Windows Communication Foundation (WCF).
- Windows Workflow Foundation (WF).

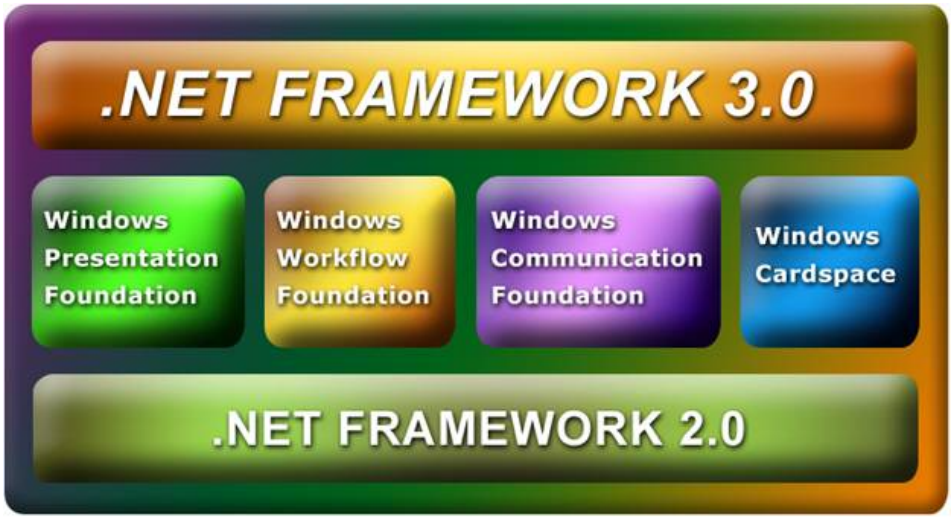


Ilustración 3: .NET Framework 3.0

Anteriormente se ha explicado qué es el Framework y qué componentes lo forman (CLR y BCL). A continuación se ofrece una breve descripción de las nuevas tecnologías añadidas por el Framework 3.0 (y que se mantienen en el actual 3.5). Puesto que el tema que nos abarca es WPF, se explicarán sin demasiado detalle algunas de las nuevas tecnologías, a excepción de esta última que será sobre la que expondrá una descripción más detallada.

3.2.3.1. Windows CardSpace

Actualmente todo el mundo posee diversas identidades en Internet (proveedor de correos, sitios de compra, chats, etc.), teniendo que recordar para cada una de ellas el nombre de usuario y la contraseña, además de la dificultad añadida de poner diferentes contraseñas en cada sitio, pues lo contrario puede ser muy peligroso. Esta tecnología provee a los usuarios la habilidad de manejar sus identidades digitales de un modo sencillo, seguro y familiar (Microsoft Corporation, 2008).

3.2.3.2. Windows Communication Foundation (WCF)

La aceptación global de los Servicios Web, los cuales incluyen protocolos estándar para la comunicación entre aplicaciones, ha cambiado el desarrollo de software. Por ejemplo las funciones que ahora incluyen los Servicios Web integran la seguridad, la coordinación de transacciones distribuidas y la comunicación fiable. Windows Communication Foundation (WCF) ha sido diseñado para ofrecer un enfoque manejable de la computación distribuida, además de una gran interoperabilidad y el apoyo directo a la orientación de los servicios (Microsoft Corporation, 2008).

En definitiva WCF es un Framework unificado para la construcción segura, confiable, transaccional e interoperable de aplicaciones distribuidas.

3.2.3.3. Windows Workflow Foundation (WF)

Es el modelo de programación, el motor y las herramientas que permiten la rápida construcción de aplicaciones “workflow enabled”. WF aumenta radicalmente la capacidad del desarrollador para modelar los procesos de negocio. Esta plataforma es altamente flexible y permite el diseño de workflows automatizados u orientados a la interacción humana y puede ser utilizado en una gran variedad de escenarios y productos (Microsoft Corporation, 2008).

3.2.3.4. Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF), es una de las novedosas tecnologías de Microsoft y uno de los pilares de Windows Vista, aunque su uso no es exclusivo de este Sistema Operativo. WPF encierra un trabajo monumental que parte de la gran experiencia acumulada en el desarrollo de interfaces de usuario desde la aparición de Windows. WPF potencia las capacidades de desarrollo de interfaces de interacción integrando y ampliando las mejores características de las aplicaciones Windows y de las aplicaciones Web (Microsoft Corporation, 2008).

Habitualmente el personal técnico se preocupa principalmente por la tecnología, los profesionales del software se centran en el modo de funcionamiento de las aplicaciones mientras que el usuario le da una gran importancia a las interfaces. La interfaz de una aplicación constituye una parte fundamental de la experiencia global del usuario con el software. La experiencia mejorada de los usuarios mediante una interfaz optimizada puede contribuir al incremento de la productividad, a la generación de clientes leales y a una ampliación de las ventas en línea, entre otras muchas.

WPF permite crear interfaces eficaces y atractivas, las cuales puedan incorporar documentos, componentes multimedia, gráficos bidimensionales y tridimensionales, animaciones, características tipo web, etc. Anteriormente para crear interfaces de usuario que ofrecieran todas las características mencionadas eran necesarias varias tecnologías, hecho que dificultaba considerablemente el proceso de creación de una interfaz. Sin embargo, al proporcionar una amplia gama de funciones en una sola tecnología, se simplifica de forma significativa la creación de interfaces de usuario modernas y más completas.

WPF es toda una plataforma que da soporte para poder colocar en las interfaces de usuario una amplia riqueza visual e interactiva e integrarlas fácilmente con la lógica de negocio. Los elementos visuales que de las IU gráficas podrán tener ahora transparencia, brillo, tonalidades, reflejo, efectos tridimensionales y animaciones. Se podrán desplegar documentos y colocar en las propias aplicaciones audio y video.

Además tendrán capacidad de enlace y navegación basado en el modelo de páginas Web. Con WPF es posible definir estilos y plantillas propios para propiciar la reutilización, aumentar la productividad de desarrollo y dar una imagen personalizada a las aplicaciones. Además con la arquitectura abierta de WPF se permite definir e implementar controles propios y componentes personalizados.

3.2.3.4.1. Diseñadores gráficos y desarrolladores

Para crear una interfaz de usuario eficaz son necesarios conocimientos que muchos profesionales del software no poseen, lo que implica la necesidad de trabajar con diseñadores de interfaces. Hasta la llegada de WPF el trabajo conjunto entre desarrolladores y diseñadores había sido bastante problemático ya que para el desarrollador puede ser muy complicado, o incluso imposible, implementar las, a priori, sencillas ideas del diseñador (Montesinos, 2006).

Las limitaciones tecnológicas, la falta de conocimiento, los malentendidos o simplemente los desacuerdos entre ambos pueden ser algunos de los motivos por los que el desarrollador no refleje la visión del diseñador. Por tanto era necesario un método de trabajo mejorado que permitiera la colaboración estrecha de ambos sin que se viese comprometido el nivel de calidad de la interfaz. Con este objetivo *WPF* incluye el lenguaje *XAML*, mediante el cual es posible definir elementos *XML*, como *Button*, *TextBox*, *Label*, etc., para especificar exactamente la apariencia de las interfaces de usuario. En la siguiente imagen se muestra el proceso de colaboración entre diseñador y desarrollador:

Expression Interactive Designer ***Visual Studio***

- Diseñador - **Ilustración 4: Interfaz y lógica de negocio** ***- Desarrollador -***

El **diseñador** utiliza una herramienta gráfica, como por ejemplo *Expression Interactive Designer*, para definir la apariencia y el modo de interacción de una interfaz de usuario. La herramienta genera automáticamente una descripción de la interfaz en XAML. A continuación el **desarrollador** importa la descripción XAML en una herramienta como *Microsoft Visual Studio* donde escribirá el código de la interfaz, como los controladores de eventos y todas las demás funciones que requiera la aplicación.

3.3. XAML

Como se ha comentado, el principal objetivo de XAML es describir gráficamente las interfaces de usuario permitiendo trabajar a diseñadores y desarrolladores juntos, pues con la llegada de estas nuevas tecnologías es posible separar la definición de la interfaz gráfica (expresada en XAML) de su funcionalidad (escrita en C# o VisualBasic.Net).

A continuación se explicará con más detalle esta innovadora tecnología que ya, está revolucionando el mundo del desarrollo software (Microsoft Corporation, 2007).

Es un lenguaje declarativo basado en XML, que simplifica en gran medida la creación de Interfaces de Usuario (IU) para el Framework 3.0 de .Net. Es posible crear elementos visibles de IU mediante el marcado declarativo XAML y separar la definición de la lógica de la aplicación, expresada mediante archivos de código subyacente (también llamados code-behind file).

XAML expresa declarativamente la creación de estructuras arbóreas de objetos .Net. Con WPF como marco de trabajo, estos objetos podrán ser desplegados visualmente e interactuarán con los usuarios permitiendo conformar una rica y apasionante interfaz de aplicación. A la vez se podrán conectar con el resto de la lógica para asociar a esta apariencia de presentación una funcionalidad todo lo compleja que se requiera.

El hecho de que se puede crear por un lado la interfaz de usuario, y por otro la lógica, XAML y WPF nos ofrece un adecuado soporte para desarrollar aplicaciones con la arquitectura conocida como MVC (Modelo-Vista-Controlador). Esta arquitectura consta de tres capas de responsabilidades:

- Capa Modelo: describe las entidades propias que se encuentran en el dominio del problema que resuelve la aplicación. Son los llamados objetos de negocio. Esta capa será implementada mediante C# o VisualBasic .Net.

- Capa Vista: define la apariencia y funcionalidad de la IU. Las enormes capacidades de WPF sirven de soporte a esta capa. La especificación de esta capa se hará mediante el lenguaje XAML.

- Capa Controlador: su propósito es crear un puente, establecer los enlaces entre las capas Modelo y Vista.

Debido a la separación en capas, el patrón MVC permite lograr mayor productividad de desarrollo y mayor flexibilidad en el producto.

Observando el código fuente de un fichero XAML se puede ver que no es otra cosa más que un fichero XML bien formado, con ciertos espacios de nombres establecidos. Sin embargo la potencia del XAML no recae estrictamente en el lenguaje, aunque éste proporcione multitud de ventajas, sino en un conjunto de clases administradas, que permiten en tiempo de compilación, convertir el XAML en una clase parcial que contendrá un código equivalente y que posteriormente será utilizada para crear los objetos.

3.3.1. Elementos XAML

Como ya se ha comentado, XAML es un lenguaje de marcado que sigue las reglas sintácticas de XML, donde cada elemento tiene un nombre y puede que varios atributos. Los elementos se definen uno dentro de otro formando una estructura arbórea, lo que le da expresividad y semántica.

XAML tiene un conjunto de reglas que mapean los **elementos** XAML **en clases** o estructuras de .Net, en particular de WPF. Los **atributos** de esos elementos son mapeados **en propiedades o eventos** de dichas clases.

```
<StackPanel>

    <Image Name=”miImagen” Heigth=”20”
    Mouse.MouseDown=”controladorDeEvento” />

</StackPanel>
```

En este ejemplo se observan dos elementos XAML: StackPanel y Image. Cada uno de ellos es mapeado a una clase definida por WPF. Al especificar la etiqueta de un elemento, se crea una instrucción para el cargador de XAML que creará una instancia de la clase correspondiente al elemento cuando el código XAML sea compilado o interpretado.

3.3.2. Propiedades y Eventos de Elementos XAML

El estado inicial de una clase que ha sido mapeada desde un elemento XAML está basada en el comportamiento definido en el constructor por defecto. Sin embargo, normalmente no se quiere utilizar una instancia de clase totalmente por defecto, por tanto se han de añadir propiedades al elemento a mapear para modificar tal comportamiento.

En el ejemplo desplegado anteriormente, podemos observar cómo se ha asignado un determinado valor a la propiedad Heigth de la imagen, así cómo también hemos definido un nuevo evento, en este caso de ratón, sobre el propio control.

En el mismo ejemplo, en el caso del control de tipo *StackPanel*, implícitamente y con el simple hecho de poner la imagen dentro de su ámbito, se estará añadiendo la imagen a la propiedad *Content* del propio control.

3.3.3. Propiedades adjuntas

En el apartado anterior se acaba de exponer que una clase por defecto de cualquier tipo de control .NET puede ser modificada mediante la adición de propiedades o eventos. A través de las propiedades adjuntas (attached property) también puede ser posible.

Hablamos de propiedad adjunta (o de dependencia), cuando se le da valor a una propiedad no dentro de la sintaxis del elemento propietario sino desde otro elemento. A continuación podemos ver un ejemplo de ello:

```
<Grid>
        <Button Grid.Row=”1”>Botón 1</Button>
</Gris>
```

Dentro del botón se hace referencia a Grid.Row, es decir, a la propiedad Row del elemento Grid dándole como valor la cadena "1". XAML interpreta esto como una llamada al método Grid.SetRow al que se le pasa como parámetros el elemento dentro del cual se referencia a la propiedad (en este caso el botón) y el valor asignado a la propiedad ("1"), colocando el botón en la primera fila del *Grid*.

Esta forma de referirse a algunas propiedades es una facilidad sintáctica que ofrece XAML y que simplifica la notación y ayuda a ganar en flexibilidad. Es muy utilizada en elementos que distribuyen otros elementos, como los paneles por ejemplo.

3.3.4. Conversores de Tipo

En XAML los valores de los atributos se expresan siempre como un string, pero no siempre los valores con los que realmente se corresponden en .Net son de tipo string. Para esto, XAML se apoya en el mecanismo de convertidores de tipo. Como su nombre indica, permiten convertir valores de un tipo a otro, generalmente valores de tipo string a valores de algún otro tipo propio de .Net.

Por ejemplo en el anterior ejemplo, se asigna el string "20" a la propiedad Heigth del elemento Image, cuando realmente esta propiedad es de tipo entero en el elemento .NET. El convertidor de tipo transforma la cadena "20" en el número entero 20 y es éste valor el que se le asigna a la propiedad.

3.3.5. Extensiones de marcado

La combinación de convertidores de tipo y las propiedades posibilita la inicialización de la mayoría de éstas. Hasta ahora se ha visto que lo que se asigna a una propiedad en XAML es una cadena (que podrá ser convertida al objeto correspondiente). Esto da como resultado valores constantes o estructuras fijas, es decir, valores conocidos en tiempo de diseño y compilación. Sin embargo, sería deseable que XAML ofreciera mayor flexibilidad a modo de poder asociar a una propiedad un valor

que no se conozca en tiempo de diseño y compilación. Por ejemplo asociar a una propiedad un valor que dependa de otra propiedad y que pueda variar en tiempo de ejecución. Para esto XAML propone lo que se conoce como extensiones de marcado (markup extensions).

El compilador XAML reconoce que se está usando una extensión de marcado cuando la cadena que contiene el valor a asignar a una propiedad está encerrada entre llaves { }.

Un ejemplo de extensión de marcado muy utilizada es Binding, la cual se utiliza para hacer enlace con datos:

```
<Label>Fuente de datos:</Label>
<TextBox Name=”fuenteDatos” Text=”Valor Inicial” />

<Label>Destino de datos:</Label>
<TextBox Text=”{Binding Text, ElementName=fuenteDatos}” />
```

3.3.6. El code-behind

XAML da soporte al concepto code-behind. Todo archivo XAML tiene un correspondiente archivo con código ejecutable. La idea es que el XAML dé la apariencia y estructura de la interfaz de usuario, mientras que el archivo code-behind debe dar el comportamiento, relacionado con la lógica de negocio.

3.3.7. XAML y SilverLight

A pesar de que XAML fue desarrollado para el uso en la plataforma Windows, actualmente es posible utilizarlo en otras plataformas gracias a SilverLight.

Silverlight es una nueva tecnología de presentación Web creada para su ejecución en distintas plataformas. Hace posible un uso más completo y atractivo de los

recursos visuales e interactivos, pudiéndose ejecutar en todos los entornos, en múltiples dispositivos y sistemas operativos de escritorio.

3.4. Ejemplo de controles predefinidos de WPF

A continuación se presentan una serie de controles usualmente utilizados en las aplicaciones construidas con WPF. Dichos controles estarán agrupados lógicamente en varias categorías, apareciendo en las mismas los más usuales de cada una de ellas. Los mismos podrán ser de muy diferentes tipologías, desde un simple botón, hasta un elemento multimedia que reproduzca video y audio.

Por cada control se podrán exponer diferentes elementos (Microsoft Corporation, 2007):

- Jerarquía de clases involucradas: se expondrá el árbol de clases vinculado con la clase de un determinado control. Normalmente un control tiene una gran variedad de antecesores, en cuanto a herencia se refiere. Por ello, un control que herede de determinada clase, dispondrá de los métodos, *propiedades* y *eventos* de dicha clase (recursivamente hasta la raíz).
- Propiedades asociadas: se expondrán las propiedades asociadas directamente a un determinado control, excluyendo las propiedades que se adquieren como consecuencia de la herencia.
- Eventos asociados: se expondrán los eventos asociados directamente a un determinado control, excluyendo los mismos que se adquieren como consecuencia de la herencia.
- Imagen del control: se expondrá una posible imagen del control en cuestión.

3.4.1. Presentación

Los controles de presentación o diseño se utilizan para administrar el tamaño, las dimensiones, la posición y la organización de los elementos secundarios.

- Border: dibuja un borde, un fondo o ambos alrededor de otro elemento.

  - La *jerarquía* de clases involucradas es:

    Jerarquía de herencia
    System.Object
    System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
    System.Windows.Media.Visual
    System.Windows.UIElement
    System.Windows.FrameworkElement
    System.Windows.Controls.Decorator
    System.Windows.Controls.Border

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **Background:** fondo del control.
    - **BorderBrush:** color del borde del control.
    - **BorderThickness:** grosor del borde del control.
    - **CornerRadius:** grado de curva de cada esquina del control.
    - **Padding:** valor de *Thickness* que describe la cantidad de espacio que hay entre el *Border* y su elemento secundario.

  - A continuación se muestra una posible *imagen* del control:

Ilustración 5: imagen Border

- Canvas: define un área en la que pueden colocarse explícitamente los elementos secundarios utilizando las coordenadas relativas al área del control.

  - La *jerarquía* de clases involucradas es:

    Jerarquía de herencia
    System.Object
    System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
    System.Windows.Media.Visual
    System.Windows.UIElement
    System.Windows.FrameworkElement
    System.Windows.Controls.Panel
    System.Windows.Controls.Canvas

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **Top:** valor que representa la distancia entre la parte superior de un elemento y la parte superior de su control *Canvas* primario.
    - **Left:** valor que representa la distancia entre la parte izquierda de un elemento y la parte izquierda de su control *Canvas* primario.
    - **Bottom:** valor que representa la distancia entre la parte inferior de un elemento y la parte inferior de su control *Canvas* primario.
    - **Right:** valor que representa la distancia entre la parte derecha de un elemento y la parte derecha de su control *Canvas* primario.

- Expander: representa un control que muestra un encabezado con una ventana contraíble en la que se muestra contenido.

  - La *jerarquía* de clases involucradas es:

    Jerarquía de herencia
    System.Object
    System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
    System.Windows.Media.Visual
    System.Windows.UIElement
    System.Windows.FrameworkElement
    System.Windows.Controls.Control
    System.Windows.Controls.ContentControl
    System.Windows.Controls.HeaderedContentControl
    System.Windows.Controls.Expander

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **ExpandDirection:** establece la dirección en la que se abre la ventana de contenido.
    - **IsExpanded:** establece si el control aparece expandido o colapsado.

  - Los *eventos* asociados son:
    - **Collapsed:** se produce cuando se cierra la ventana de contenido de un control *Expander* y sólo *Header* está visible.
    - **Expanded:** se produce cuando se abre la ventana de contenido de un control *Expander* para mostrar su encabezado y contenido.

  - A continuación se muestra una posible *imagen* del control:

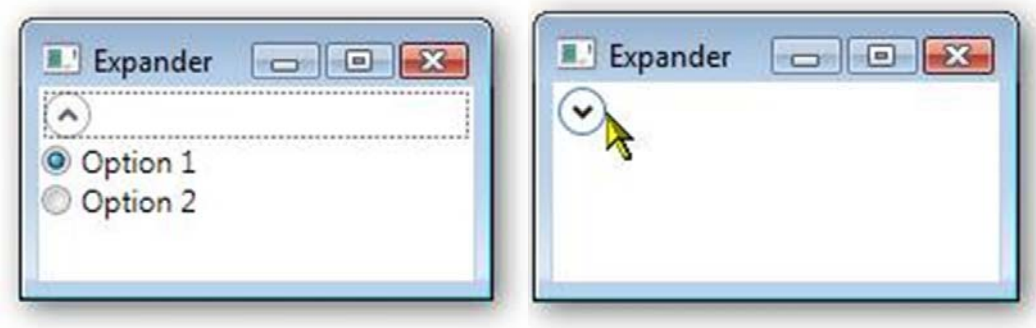
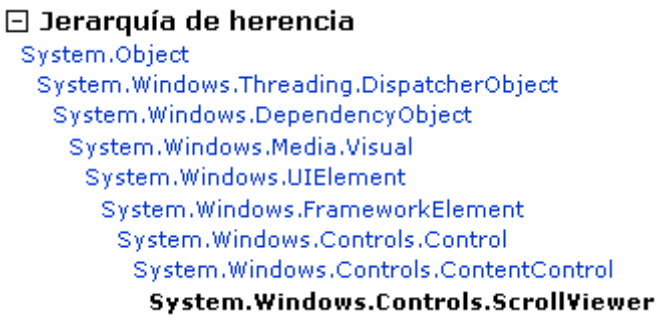


Ilustración 6: imagen Expander

- ScrollViewer: representa un área desplazable que puede contener otros elementos visibles.
  - La *jerarquía* de clases involucradas es:



  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **ComputedHorizontalScrollBarVisibility:** obtiene un valor que indica si la *ScrollBar* horizontal se encuentra visible.
    - **ComputedVerticalScrollBarVisibility:** obtiene un valor que indica si la *ScrollBar* vertical se encuentra visible.
    - **ContentHorizontalOffset:** obtiene el desplazamiento horizontal del contenido visible.
    - **HorizontalOffset:** obtiene el desplazamiento horizontal del contenido desplazado.
    - **ContentVerticalOffset:** obtiene el desplazamiento vertical del contenido visible.
    - **VerticalOffset:** obtiene el desplazamiento vertical del contenido desplazado.
    - **HorizontalScrollBarVisibility:** valor que indica si debe mostrarse una *ScrollBar* horizontal.
    - **VerticalScrollBarVisibility:** valor que indica si debe mostrarse una *ScrollBar* vertical.
    - **ScrollableHeight:** valor que representa el tamaño vertical del elemento de contenido que puede desplazarse.
    - **ScrollableWidth:** valor que representa el tamaño horizontal del elemento de contenido que puede desplazarse.
    - **ViewportHeight:** tamaño vertical de la ventanilla de contenido.

- **ViewportWidth:** tamaño horizontal de la ventanilla de contenido.

· Los *eventos* relacionados son:

- **ScrollChanged:** se produce cuando se detectan cambios en la extensión, tamaño de ventanilla o posición de desplazamiento.

· A continuación se muestra una posible *imagen* del control:

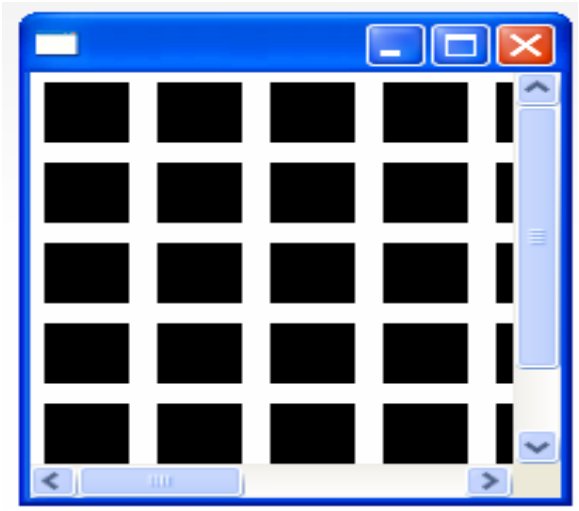


Ilustración 7: imagen ScrollViewer

- StackPanel: organiza los elementos secundarios en una única línea que se puede orientar horizontal o verticalmente.

· La *jerarquía* de clases involucradas es:

· Las *propiedades* asociadas que más se suelen utilizar son:

- **CanHorizontallyScroll:** indica si *StackPanel* puede desplazarse en la dimensión horizontal.
- **CanVerticallyScroll:** indica si *StackPanel* puede desplazarse en la dimensión vertical.
- **HorizontalOffset:** contiene el desplazamiento horizontal del contenido desplazado.
- **VerticalOffset:** contiene el desplazamiento vertical del contenido desplazado.
- **Orientation:** indica la dimensión según la cual se apilan los elementos secundarios.
- **ScrollOwner:** identifica el contenedor que controla el comportamiento del desplazamiento en *StackPanel*.
- **ViewportHeight:** tamaño vertical de la ventanilla de contenido.
- **ViewportWidth:** tamaño horizontal de la ventanilla de contenido.

- Window: permite crear, configurar, mostrar y administrar la duración de las ventanas y los cuadros de diálogo.

  - La *jerarquía* de clases involucradas es:

    Jerarquía de herencia
    System.Object
    System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
    System.Windows.Media.Visual
    System.Windows.UIElement
    System.Windows.FrameworkElement
    System.Windows.Controls.Control
    System.Windows.Controls.ContentControl
    System.Windows.Window

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **AllowsTransparency:** indica si el área cliente de una ventana admite transparencia.
    - **Icon:** icono asociado a una ventana.
    - **IsActive:** indica si la ventana está activa.

- **Left:** posición del borde izquierdo de la ventana con respecto al escritorio.
- **OwnedWindows:** colección de ventanas de las que esta ventana es el propietario.
- **Owner:** objeto *Window* que es la ventana propietaria de este objeto *Window*.
- **ResizeMode:** modo de cambio de tamaño.
- **RestoreBounds:** obtiene el tamaño y la ubicación de una ventana antes de ser minimizada o maximizada.
- **ShowActivated:** valor que indica si una ventana se activa la primera vez que se muestra.
- **ShowInTaskbar:** valor que indica si la ventana tiene un botón de barra de tareas.
- **SizeToContent:** valor que indica si una ventana ajustará automáticamente su tamaño al de su contenido.
- **Title:** título de una ventana.
- **Top:** posición del borde superior de la ventana con respecto al escritorio.
- **WindowStartupLocation:** posición de la ventana cuando se muestra por primera vez.
- **WindowState:** valor que indica si una ventana está restaurada, minimizada o maximizada.
- **WindowStyle:** estilo de borde de una ventana.

- Los *eventos* relacionados son:
  - **Activated:** se produce cuando una ventana se convierte en la ventana en primer plano.
  - **Closed:** se produce cuando la ventana está a punto de cerrarse.
  - **Closing:** se produce inmediatamente después de llamar a *Close* y se puede controlar para que se cancele el cierre de la ventana.
  - **ContentRendered:** se produce una vez representado el contenido de una ventana.

- **Deactivated:** se produce cuando una ventana se sitúa en segundo plano.
- **LocationChanged:** se produce cuando cambia la ubicación de una ventana.
- **StateChanged:** se produce cuando cambia la propiedad *WindowState* de la ventana.

· A continuación se muestra una posible *imagen* del control:

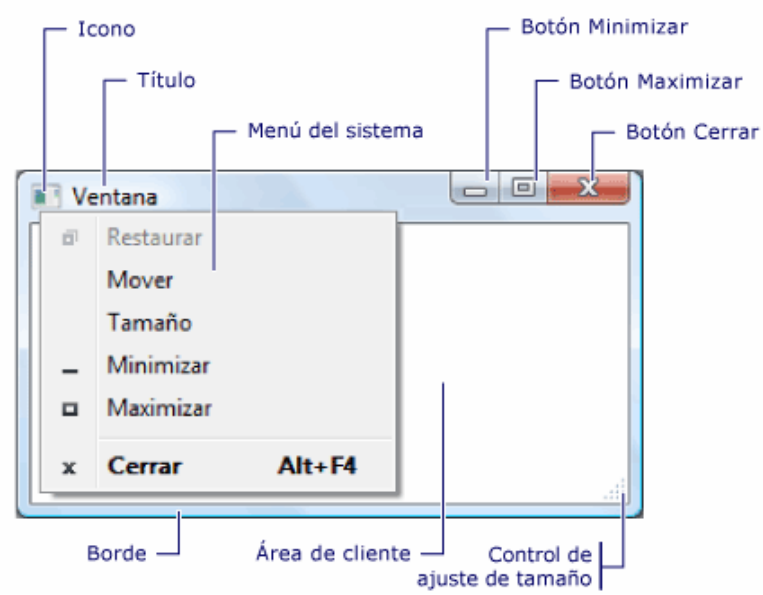


Ilustración 8: imagen Window

3.4.2. Botones

El botón es uno de los controles de interfaz de usuario más básicos. Las aplicaciones suelen realizar algún tipo de tarea en el evento *Click* cuando un usuario hace clic en un botón.

- Button: representa un control de botón de Windows, que reacciona al evento ButtonBase.Click.

  - La *jerarquía* de clases involucradas es:

    Jerarquía de herencia
    System.Object
    System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
    System.Windows.Media.Visual
    System.Windows.UIElement
    System.Windows.FrameworkElement
    System.Windows.Controls.Control
    System.Windows.Controls.ContentControl
    System.Windows.Controls.Primitives.ButtonBase
    System.Windows.Controls.Button

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **IsCancel:** valor que indica si un *Button* es un botón Cancelar. Un usuario puede activar el botón Cancelar presionando la tecla ESC.
    - **IsDefault:** indica si un *Button* es el botón predeterminado. Un usuario invoca el botón predeterminado presionando la tecla ENTRAR.

  - Los *eventos* relacionados son:
    - **Click:** se produce cuando se hace clic en el botón.

  - A continuación se muestra una posible *imagen* del control:

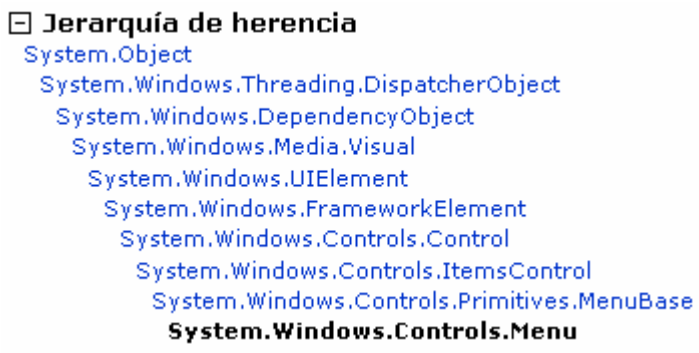
Ilustración 9: imagen Button

3.4.3. Menús

Los menús se utilizan para agrupar acciones relacionadas o proporcionar ayuda contextual.

- Menu: representa un control de menú de Windows que le permite organizar jerárquicamente los elementos asociados a comandos y controladores de eventos.

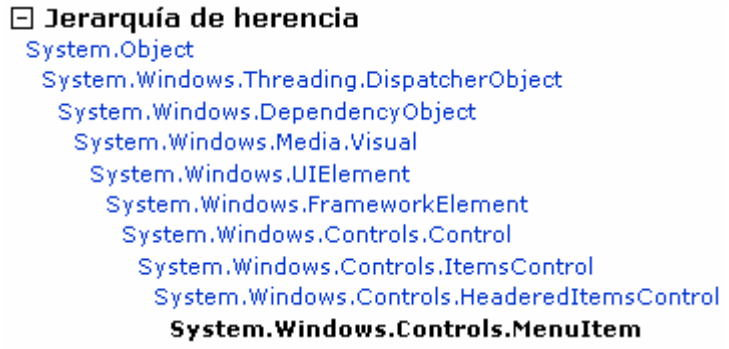
  - La *jerarquía* de clases involucradas es:

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **IsMainMenu:** indica si este *Menu* recibe una notificación de activación de menú principal.

- MenuItem: representa un elemento seleccionable dentro de un control de tipo *Menu.*

  - La *jerarquía* de clases involucradas es:



- Las *propiedades* asociadas que más se suelen utilizar son:
  - **Command:** comando asociado al elemento de menú.
  - **CommandParameter:** parámetro que se va a pasar a la propiedad *Command* de *MenuItem.*
  - **CommandTarget:** elemento de destino en el que se produce el comando especificado.
  - **IsCheckable:** valor que indica si se puede activar un *MenuItem*.
  - **IsChecked:** valor que indica si está activado un *MenuItem*.
  - **IsHighlighted:** indica si *MenuItem* aparece resaltado.
  - **IsPressed:** indica si se ha presionado el *MenuItem.*
  - **IsSubmenuOpen:** indica si el submenú de *MenuItem* está abierto.
  - **IsSuspendingPopupAnimation:** obtiene si un menú suspende las animaciones en su control *Popup.*
  - **Role:** valor que indica la función del *MenuItem.*
  - **StaysOpenOnClick:** valor que indica que el submenú en el que se encuentra *MenuItem* no se debe cerrar al hacer clic en este elemento.

- Los *eventos* relacionados son:
  - **Checked:** se produce cuando se activa un elemento de menú.
  - **Click:** Se produce cuando se hace clic en *MenuItem*.
  - **SubmenuClosed:** se produce cuando el estado de la propiedad *IsSubmenuOpen* cambia a *false*.
  - **SubmenuOpened:** se produce cuando el estado de la propiedad *IsSubmenuOpen* cambia a *true*.
  - **Unchecked:** se produce cuando se desactiva *MenuItem*.

- A continuación se muestra una posible *imagen* del control *Menu*, conteniendo diferentes controles *MenuItem*:

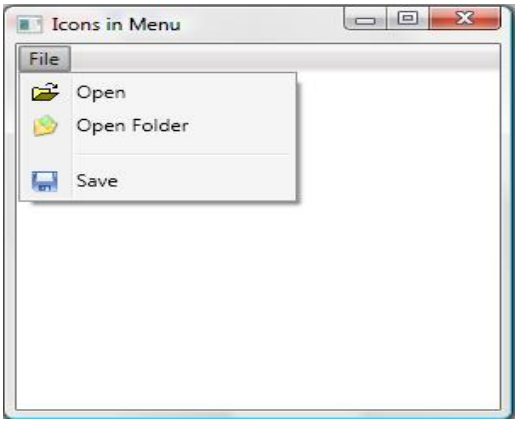


Ilustración 10: imagen Menu/MenuItem

- ToolBar: proporciona un contenedor para un grupo de comandos o controles.

- La *jerarquía* de clases involucradas es:

Jerarquía de herencia
System.Object
System.Windows.Threading.DispatcherObject
System.Windows.DependencyObject
System.Windows.Media.Visual
System.Windows.UIElement
System.Windows.FrameworkElement
System.Windows.Controls.Control
System.Windows.Controls.ItemsControl
System.Windows.Controls.HeaderedItemsControl
System.Windows.Controls.ToolBar

- Las *propiedades* asociadas que más se suelen utilizar son:
  - **Band:** valor que indica dónde debe situarse la barra de herramientas en ToolBarTray (contenedor del ToolBar).
  - **BandIndex:** número de índice de banda que indica la posición de la barra de herramientas en la banda.
  - **HasOverflowItems:** valor que indica si la barra de herramientas tiene elementos que no están visibles.

- **IsOverflowOpen:** valor que indica si el área de desbordamiento de *ToolBar* está actualmente visible.
- **Orientation:** orientación de *ToolBar*.
- **IsOverflowItem:** valor que indica si el elemento *ToolBar* es un elemento de desbordamiento.
- **OverflowMode:** indica cuándo hay que colocar un elemento en el panel de desbordamiento en lugar de en el panel principal.

- A continuación se muestra una posible *imagen* del control:

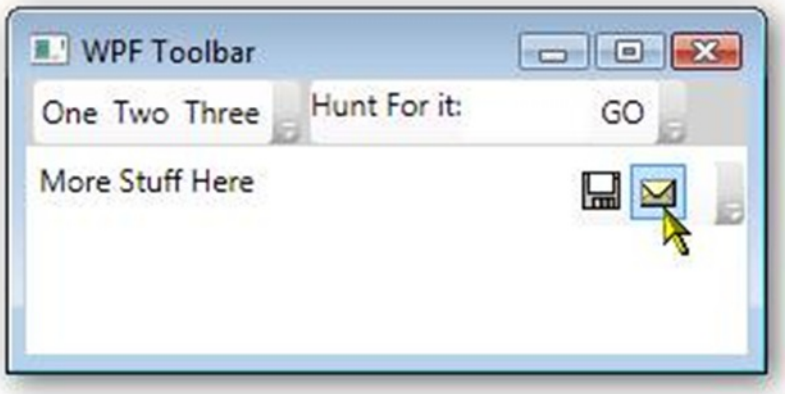


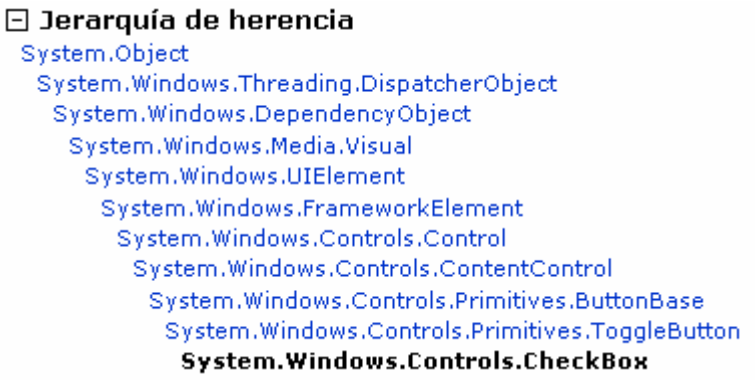
Ilustración 11: imagen ToolBar

3.4.4. Selección

Los controles de selección se utilizan para permitir al usuario seleccionar una o más opciones.

- CheckBox: representa un control que un usuario puede activar y desactivar.

  - La *jerarquía* de clases involucradas es:



- A continuación se muestra una posible *imagen* del control:

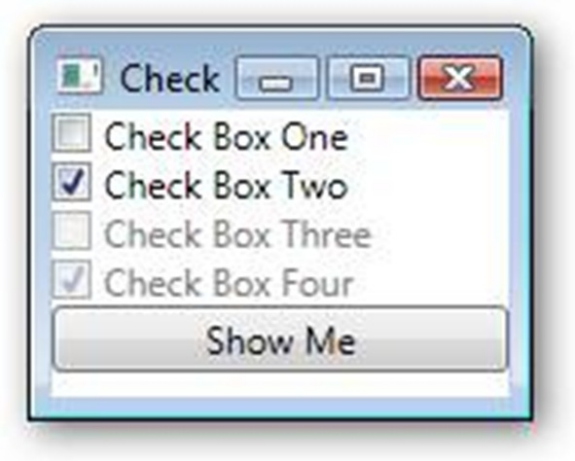
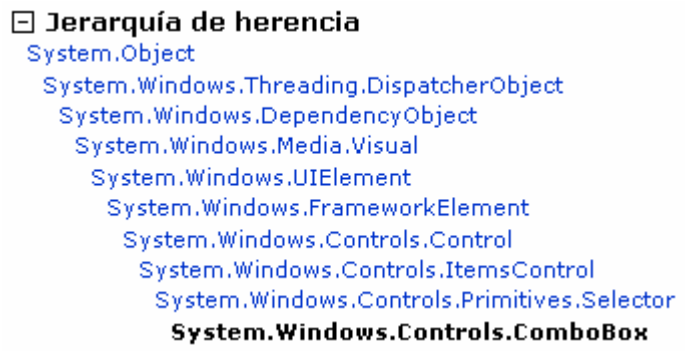


Ilustración 12: imagen CheckBox

- ComboBox: representa un control de selección con una lista desplegable que se puede mostrar u ocultar haciendo clic en la flecha del control.

  - La *jerarquía* de clases involucradas es:



  - Las *propiedades* asociadas que más se suelen utilizar son:

- **IsDropDownOpen:** valor que indica si el área desplegable de un cuadro combinado está abierta en la actualidad.
- **IsEditable:** obtiene o establece un valor que habilita o deshabilita la edición de texto en el cuadro de texto de *ComboBox*.
- **IsReadOnly:** valor que habilita el modo de sólo selección, donde el contenido del cuadro combinado se puede seleccionar pero no modificar.
- **StaysOpenOnEdit:** obtiene o establece si un objeto *ComboBox* que se abre y muestra un control desplegable permanecerá abierto cuando un usuario haga clic en *TextBox*.
- **Text:** texto del elemento que está seleccionado en la actualidad.

· Los *eventos* relacionados son:

- **DropDownClosed:** se produce cuando la lista desplegable del cuadro combinado se cierra.
- **DropDownOpened:** se produce cuando la lista desplegable del cuadro combinado se abre.

· A continuación se muestra una posible *imagen* del control:

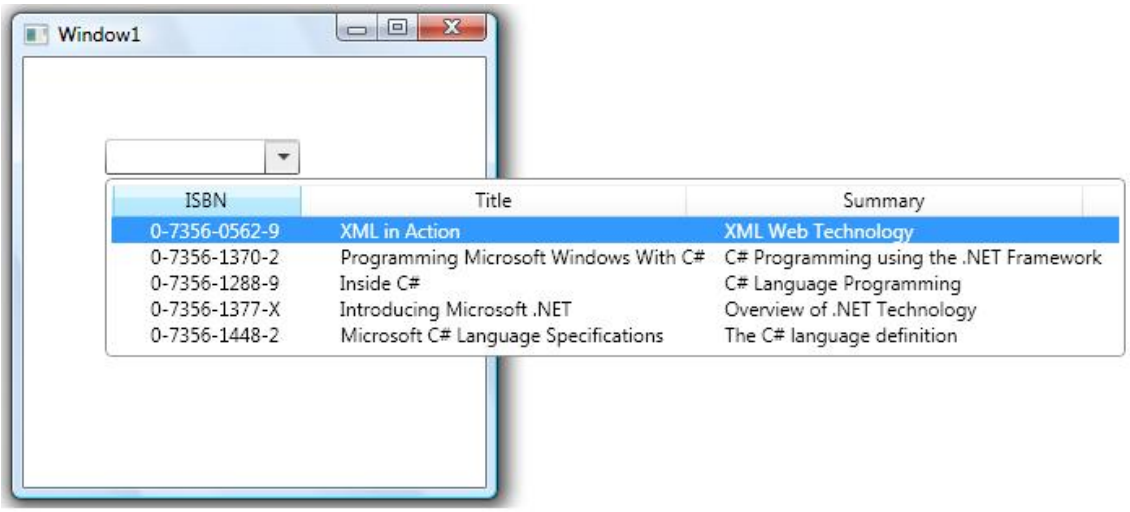
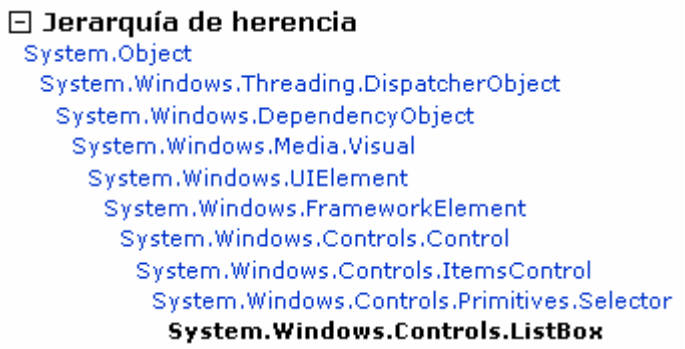


Ilustración 13: Imagen ComboBox

- ListBox: contiene una lista de elementos seleccionables.

  - La *jerarquía* de clases involucradas es:



  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **SelectionMode:** comportamiento de la selección para ListBox.

  - A continuación se muestra una posible *imagen* del control:

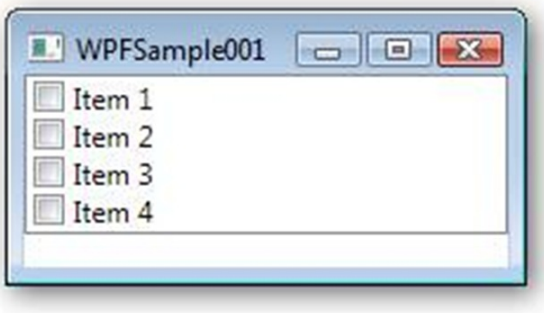


Ilustración 14: imagen ListBox

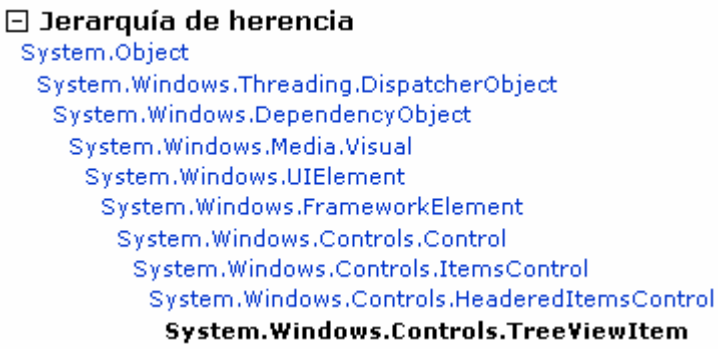
- TreeView: representa un control que muestra datos jerárquicos en una estructura de árbol con elementos que se pueden expandir y contraer.

  - La *jerarquía* de clases involucradas es:

- Las *propiedades* asociadas que más se suelen utilizar son:
  - **SelectedItem:** elemento seleccionado en el *TreeView*.
  - **SelectedValue:** valor de la propiedad que es la especificada por *SelectedValuePath* para *SelectedItem.*
  - **SelectedValuePath:** ruta de acceso que se utiliza para obtener *SelectedValue* de *SelectedItem* en *TreeView*.
  - **SelectedItemChanged:** se produce cuando *SelectedItem* cambia.

- TreeViewItem: implementa un elemento seleccionable de un control *TreeView*.

  - La *jerarquía* de clases involucradas es:



  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **IsExpanded:** valor que indica si los elementos anidados en *TreeViewItem* están expandidos o contraídos.
    - **IsSelected:** obtiene o establece si un control *TreeViewItem* está seleccionado.
    - **IsSelectionActive:** valor que indica si el *TreeViewItem* tiene el foco del teclado.

  - Los *eventos* relacionados son:
    - **Collapsed:** se produce cuando la propiedad *IsExpanded* cambia de *true* a *false*.
    - **Expanded:** se produce cuando la propiedad *IsExpanded* cambia de *false* a *true.*

- **Selected:** se produce cuando la propiedad *IsSelected* de *TreeViewItem* cambia de *false* a *true*.
- **Unselected:** se produce cuando la propiedad *IsSelected* de *TreeViewItem* cambia de *true* a *false*

· A continuación se muestra una posible *imagen* del control *TreeView*, el cual contiene una serie de elementos *TreeViewItem*:

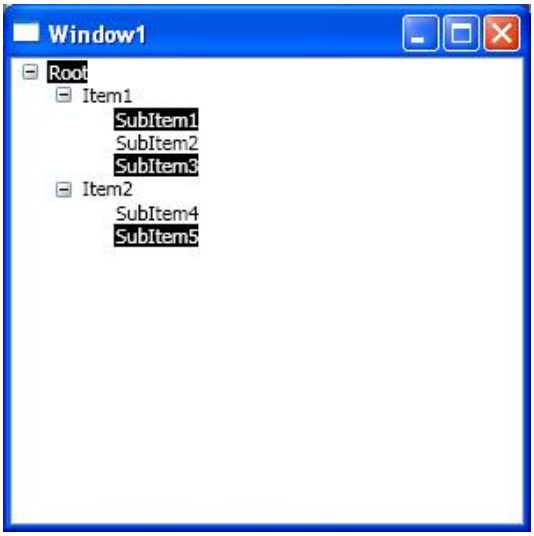


Ilustración 15: TreeView/TreeViewItem

- RadioButton: representa un botón que el usuario puede seleccionar, pero no borrar.

· La *jerarquía* de clases involucradas es:

Jerarquía de herencia
System.Object
System.Windows.Threading.DispatcherObject
System.Windows.DependencyObject
System.Windows.Media.Visual
System.Windows.UIElement
System.Windows.FrameworkElement
System.Windows.Controls.Control
System.Windows.Controls.ContentControl
System.Windows.Controls.Primitives.ButtonBase
System.Windows.Controls.Primitives.ToggleButton
System.Windows.Controls.RadioButton

- Las *propiedades* asociadas que más se suelen utilizar son:
  - **GroupName:** nombre que especifica qué controles *RadioButton* se excluyen mutuamente.

- A continuación se muestra una posible *imagen* del control:

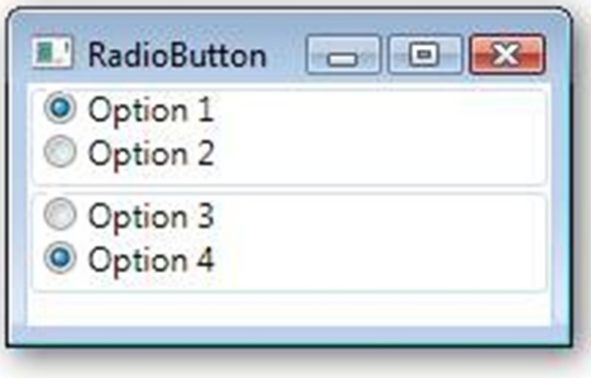


Ilustración 16: imagen RadioButton

- Slider: representa un control que permite al usuario seleccionar a partir de un intervalo de valores moviendo un control *Thumb* a lo largo de *Track*.

- La *jerarquía* de clases involucradas es:

Jerarquía de herencia
System.Object
System.Windows.Threading.DispatcherObject
System.Windows.DependencyObject
System.Windows.Media.Visual
System.Windows.UIElement
System.Windows.FrameworkElement
System.Windows.Controls.Control
System.Windows.Controls.Primitives.RangeBase
System.Windows.Controls.Slider

- Las *propiedades* asociadas que más se suelen utilizar son:
  - **AutoToolTipPlacement:** obtiene o establece si se muestra una información sobre herramientas que contiene el valor actual de *Slider* al presionar *Thumb*.
  - **AutoToolTipPrecision:** obtiene o establece el número de dígitos que se muestran en el lado derecho del separador de decimales para el valor *Value* del *Slider* en una información sobre herramientas.

- **DecreaseLarge:** obtiene un comando que disminuye el valor de *Slider* en la misma cantidad que la propiedad *LargeChange*.
- **DecreaseSmall:** obtiene un comando que disminuye el valor de *Slider* en la misma cantidad que la propiedad *SmallChange*.
- **Delay:** obtiene o establece la cantidad de tiempo en milisegundos que espera *RepeatButton*, mientras se presiona, antes de que se ejecute un comando que mueva *Thumb*, como un comando *DecreaseLarge*.
- **IncreaseLarge:** obtiene un comando que aumenta el valor de *Slider* en la misma cantidad que la propiedad *LargeChange*.
- **IncreaseSmall:** obtiene un comando que aumenta el valor de *Slider* en la misma cantidad que la propiedad *SmallChange*.
- **Interval:** cantidad de tiempo en milisegundos entre los comandos de aumento o disminución cuando un usuario hace clic en *RepeatButton* de *Slider*.
- **IsDirectionReversed:** dirección del valor de aumento.
- **IsMoveToPointEnabled:** valor que indica si *Thumb* de *Slider* se mueve inmediatamente a la ubicación del clic del ratón que se produce mientras el puntero del ratón hace una pausa en la pista *Slider*.
- **IsSelectionRangeEnabled:** valor que indica si *Slider* muestra un intervalo de selección a lo largo de *Slider*.
- **IsSnapToTickEnabled:** valor que indica si *Slider* mueve automáticamente *Thumb* a la marca de paso más próxima.
- **MaximizeValue:** obtiene un comando que establece el *Value* del *Slider* al valor *Maximum*.
- **MinimizeValue:** obtiene un comando que establece el *Value* del *Slider* al valor *Minimum*.
- **Orientation:** orientación de un *Slider*.
- **SelectionEnd:** obtiene o establece el valor mayor de una selección especificada para *Slider*.
- **SelectionStart:** obtiene o establece el valor menor de una selección especificada para *Slider*.

- **TickFrequency:** intervalo entre las marcas de paso.
- **TickPlacement:** obtiene o establece la posición de marcas de paso con respecto a *Track* de *Slider*.
- **Ticks:** obtiene o establece las posiciones de las marcas de paso para que se muestren para *Slider*.

· A continuación se muestra una posible *imagen* del control:

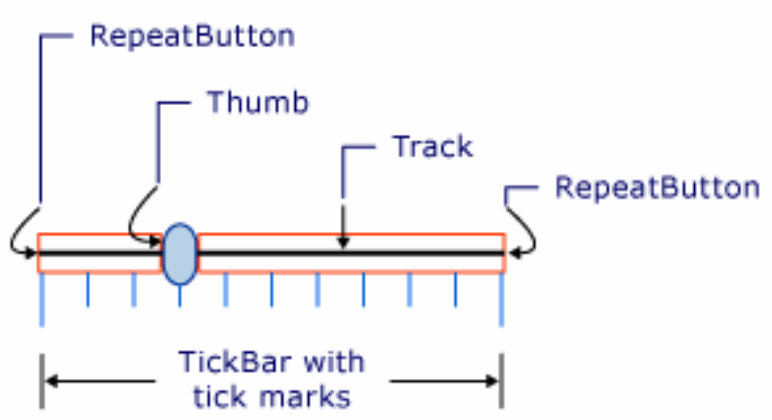


Ilustración 17: imagen Slider

3.4.5. Información del usuario

Los controles de información del usuario proporcionan comentarios contextuales o aclaraciones sobre la interfaz de usuario de una aplicación. Normalmente, el usuario no puede interactuar con estos controles.

- Label: representa la etiqueta de texto de un control y proporciona compatibilidad con las teclas de acceso.

- La *jerarquía* de clases involucradas es:

- Las *propiedades* asociadas que más se suelen utilizar son:
  - **Target:** obtiene o establece el elemento que recibe el foco cuando el usuario presiona la tecla de acceso de la etiqueta.

- ProgressBar: indica el progreso de una operación
  - La *jerarquía* de clases involucradas es:

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **IsIndeterminate:** obtiene o establece si *ProgressBar* muestra valores reales o información sobre el progreso continuo y genérico.
    - **Orientation:** orientación de una *ProgressBar (*horizontal o vertical).

- A continuación se muestra una posible *imagen* del control:

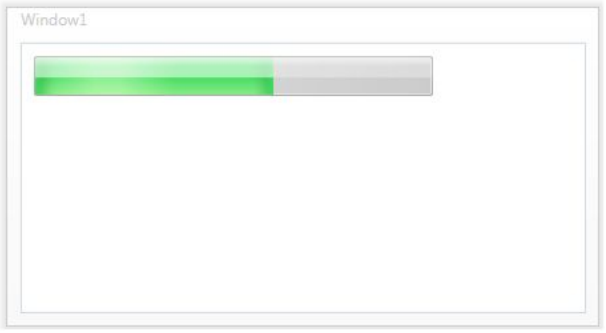


Ilustración 18: imagen ProgressBar

3.4.6. Entrada

Los controles de entrada permiten al usuario escribir texto y otros contenidos.

- TextBox: representa un control que se puede utilizar para mostrar o editar texto sin formato.
  - La *jerarquía* de clases involucradas es:

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **CaretIndex:** índice de posición de inserción.
    - **CharacterCasing:** obtiene o establece si se utilizan minúsculas o mayúsculas cuando se escriben los caracteres manualmente en el cuadro de texto.
    - **LineCount:** número total de líneas en el cuadro de texto.
    - **MaxLength:** número máximo de caracteres que se pueden escribir manualmente en el cuadro de texto.

- **MaxLines:** máximo de líneas visibles.
- **MinLines:** número mínimo de líneas visibles.
- **SelectedText:** contenido de la selección actual en el cuadro de texto.
- **SelectionLength:** valor que indica el número de caracteres de la selección actual en el cuadro de texto.
- **SelectionStart:** índice de carácter para el principio de la selección actual.
- **TextAlignment:** alineación horizontal del contenido del cuadro de texto.
- **TextDecorations:** obtiene las decoraciones de texto que se aplican al cuadro de texto.
- **TextWrapping:** modo en que debe ajustar el texto el cuadro de texto.
- **Typography:** variaciones tipográficas actualmente vigentes para el contenido del texto del cuadro de texto.

· A continuación se muestra una posible *imagen* del control:

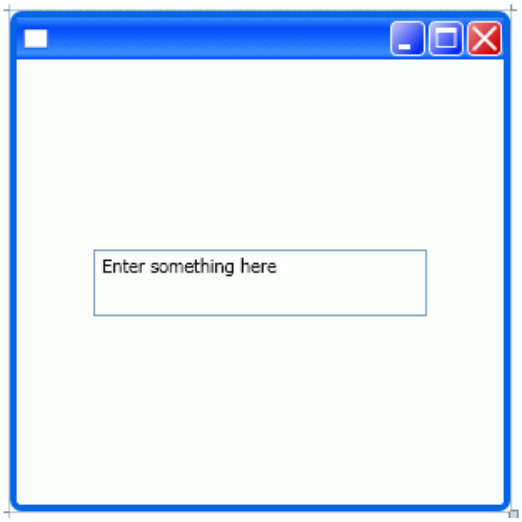
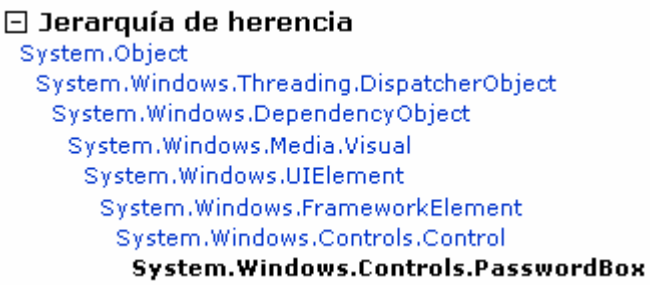


Ilustración 19: imagen TextBox

- PasswordBox: representa un control diseñado para escribir y administrar las contraseñas.
  · La *jerarquía* de clases involucradas es:



- Las *propiedades* asociadas que más se suelen utilizar son:
  - **MaxLength:** longitud máxima para las contraseñas que va a administrar este *PasswordBox*
  - **Password:** contraseña mantenida actualmente por *PasswordBox*.
  - **PasswordChar:** carácter de enmascaramiento para *PasswordBox*.
  - **SecurePassword:** obtiene la contraseña que *PasswordBox* contiene actualmente como *SecureString*.

- Los *eventos* relacionados son:
  - **PasswordChanged:** se produce cuando cambia el valor de la propiedad *Password*.

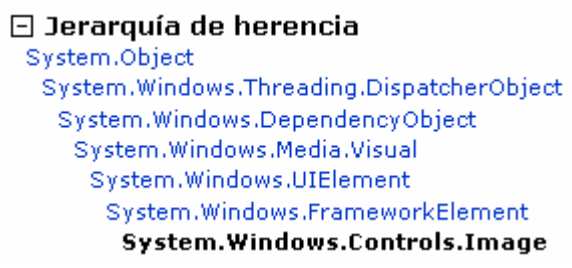
- A continuación se muestra una posible *imagen* del control:

Ilustración 20: imagen PasswordBox

3.4.7. Multimedia

WPF incluye compatibilidad integrada para hospedar contenido de audio y vídeo, así como códecs para la mayoría de los formatos de imagen más populares.

- Image: representa un control que muestra una imagen.
  - La *jerarquía* de clases involucradas es:



- Las *propiedades* asociadas que más se suelen utilizar son:
  - **BaseUri:** identificador de recursos uniforme (URI) base del objeto Image.
  - **Source:** obtiene o establece el objeto ImageSource de la imagen.
  - **Stretch:** valor que describe cómo debe expandirse Image para rellenar el rectángulo de destino.
  - **StretchDirection:** valor que indica cómo va a ajustarse la imagen.

- Los *eventos* relacionados son:
  - **ImageFailed:** se desencadena cuando hay un error en la imagen.

- MediaElement: representa un control que contiene audio y/o vídeo.
  - La *jerarquía* de clases involucradas es:

  - Las *propiedades* asociadas que más se suelen utilizar son:
    - **BufferingProgress:** Obtiene un valor que indica el porcentaje del progreso del almacenamiento en búfer realizado.
    - **CanPause:** valor que indica si se puede hacer una pausa en los multimedia.
    - **Clock:** Obtiene o establece el reloj asociado al *MediaTimeline* que controla la reproducción de los multimedia.

- **DownloadProgress:** valor de porcentaje que indica la cantidad de descarga completada del contenido que se encuentra en un servidor remoto.
- **HasAudio:** valor que indica si los multimedia tienen audio.
- **HasVideo:** valor que indica si los multimedia tienen video.
- **IsBuffering:** valor que indica si los multimedia están almacenando en el búfer.
- **IsMuted:** indica si el audio está desactivado.
- **LoadedBehavior:** comportamiento de carga *MediaState* para los multimedia.
- **NaturalDuration:** duración natural de los multimedia.
- **NaturalVideoHeight:** alto del vídeo asociado a los multimedia.
- **NaturalVideoWidth:** ancho del vídeo asociado a los multimedia
- **Position:** posición actual del progreso en el tiempo de reproducción de los multimedia.
- **ScrubbingEnabled:** valor que indica si *MediaElement* actualizará los fotogramas para las operaciones de búsqueda mientras está en pausa.
- **Source:** obtiene o establece un origen de multimedia en *MediaElement*.
- **SpeedRatio:** obtiene o establece la relación de velocidad de los multimedia.
- **Stretch:** valor *Stretch* que describe cómo *MediaElement* rellena el rectángulo de destino.
- **StretchDirection:** valor que determina las restricciones en la escala que se aplican a la imagen.
- **Volume:** volumen de multimedia.

- Los *eventos* relacionados son:
  - **BufferingEnded:** se produce cuando ha finalizado el almacenamiento en búfer de los multimedia.
  - **BufferingStarted:** se produce cuando ha comenzado el almacenamiento en búfer de los multimedia.

- **MediaEnded:** se produce cuando han finalizado los multimedia.
- **MediaFailed:** se produce cuando se encuentra un error.
- **MediaOpened:** se produce cuando ha finalizado la carga de los multimedia.
- **ScriptCommand:** se produce cuando se encuentra un comando de script en los multimedia.

4. Metodologías

A continuación se va a detallar información importante acerca de las metodologías, explicando qué son exactamente y la relación existente con el tema de calidad abordado anteriormente. Además, para entrar en situación se va a proceder a describir cual suele ser la estructura de las mismas mediante un ejemplo de una metodología real, enormemente utilizada.

4.1. Qué es una metodología

Una metodología se puede definir como el enfoque de un problema de manera total, organizada, sistemática y disciplinada (IRM, 2005).

Las personas a lo largo de su vida profesional obtienen experiencia relativa a la puesta en marcha de determinados procesos. El objetivo final de la labor de una determinada persona no sólo consiste saber desempeñar dicha labor, sino de hacerlo de la mejor manera posible. Por ello, cuanta más experiencia se tenga sobre un determinado proceso, se obtendrán mejores resultados con un menor esfuerzo.

El problema fundamental es que la experiencia de la que se habla se pierde con las personas. Para una empresa es bueno tener recursos humanos especializados y con años de experiencia en un determinado campo, sin embargo eso no durará toda la vida. Las metodologías contienen conocimientos y experiencia relativos a un determinado proceso que se ha ido mejorando durante años. Por ello, las personas sin ningún tipo de experiencia podrán disponer de una base sólida sobre la que comenzar, mejorándola incluso con el tiempo.

4.2. QA y Metodología de calidad

Una metodología de calidad en el ámbito de la ingeniería del software, es aquella que expone el proceso necesario para garantizar que una determinada aplicación cumple con las especificaciones de calidad. A este proceso se le suele denominar proceso de "aseguramiento de calidad" (del inglés Quality Assurance o abreviado Q.A.).

Cuando se habla de calidad en un desarrollo software, se hace referencia al grado con el que dicho sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario.

A continuación se expondrá la forma que tiene Métrica 3 de llevar a cabo los procesos de calidad sobre cualquier tipo de aplicaciones.

4.3. Calidad en Métrica3

Métrica es una metodología para la planificación, desarrollo y mantenimiento de sistemas de información. Dicha metodología (versión 3) es una iniciativa promovida por el Consejo Superior de Informática, órgano colegiado encargado de la elaboración y desarrollo de la política informática del Gobierno.

Esta ha sido concebida para abarcar el desarrollo completo de Sistemas de Información sea cual sea su complejidad y magnitud, por lo cual su estructura y los perfiles de los participantes que intervienen deberán adaptarse y dimensionarse en cada momento de acuerdo a las características particulares de cada proyecto.

4.3.1. Procesos en Métrica3

Los procesos que Métrica3 enuncia para cualquier desarrollo software son los que se enumeran a continuación:

- Estudio de Viabilidad del Sistema (EVS).
- Análisis del sistema de Información (ASI).
- Diseño del Sistema de Información (DSI).
- Construcción del Sistema de Información (CSI).
- Implantación y Aceptación del sistema (IAS).
- Mantenimiento del Sistema de Información (MSI).

Cada proceso de los mencionados anteriormente se compondrá de actividades, las cuales serán subprocesos contenedores de tareas.

4.3.2. Interfaces en Métrica3

En Métrica3, además habrá **interfaces**, las cuales definen una serie de actividades de tipo organizativo o de soporte al proceso de desarrollo. Las interfaces descritas en Métrica3 son:

- Gestión de proyectos (GP).
- Seguridad (SEG).
- **Aseguramiento de Calidad (CAL).**
- Gestión de la Configuración (GC).

Como se puede observar, hay una interfaz de aseguramiento de calidad, entre otras tantas, pero nos centraremos fundamentalmente en esta última.

A continuación se muestra un diagrama en el que aparecen todas y cada una de las fases mencionadas anteriormente, pudiendo observar que hay tantas fases del

proyecto como interfaces de calidad (puesto que toda fase requiere de comprobación de calidad):

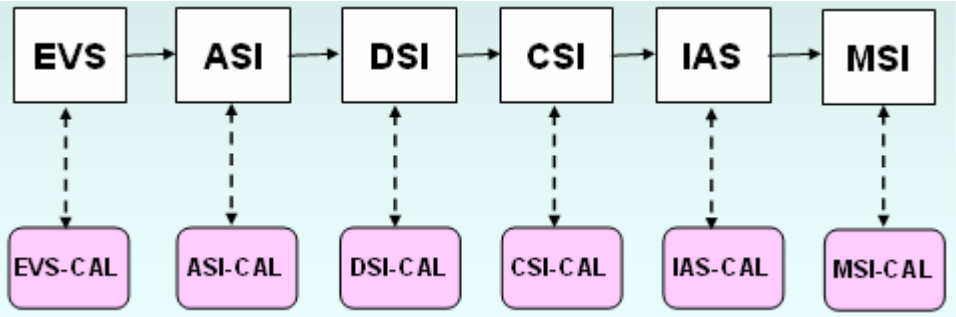


Ilustración 21: fases Métrica3

En los **diagramas detallados** expuestos a continuación se podrá observar la correspondencia entre las actividades de los diferentes procesos o fases, y las de la interfaz de Aseguramiento de la Calidad

Cuando existe una relación directa entre una actividad de un determinado proceso (EVS, ASI, etc.) respecto a la de la interfaz de calidad, significará habrá que llevar a cabo de forma paralela ambas actividades. En caso de que la relación de la actividad de la interfaz de calidad lo sea entre dos actividades de un determinado proceso, significará que la actividad de calidad deberá realizarse secuencialmente entre ambas actividades.

- Fase EVS:

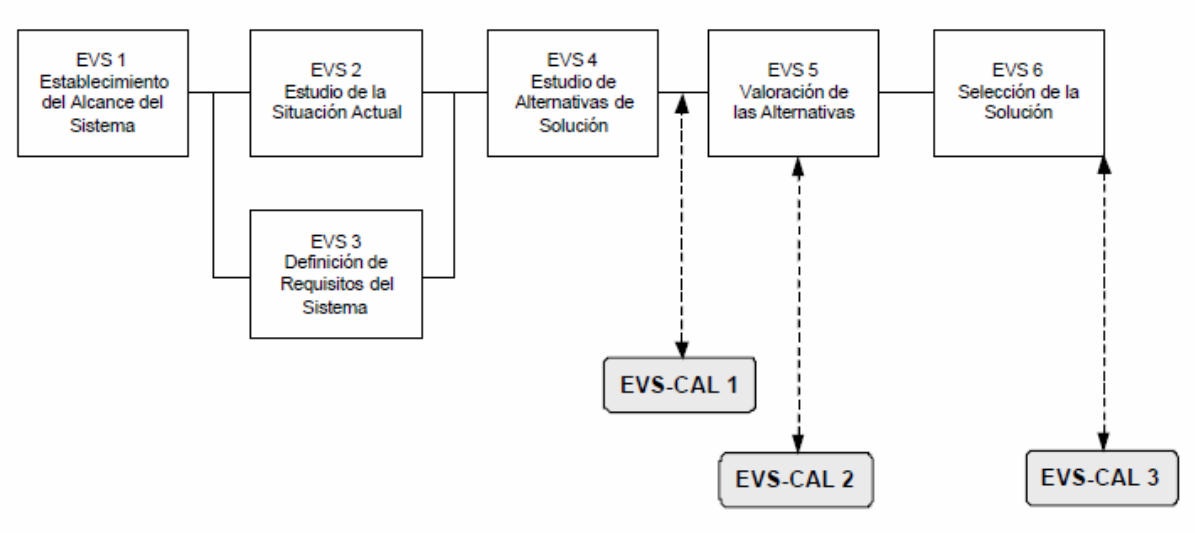


Ilustración 22: Fase EVS Métrica3

- Fase ASI:

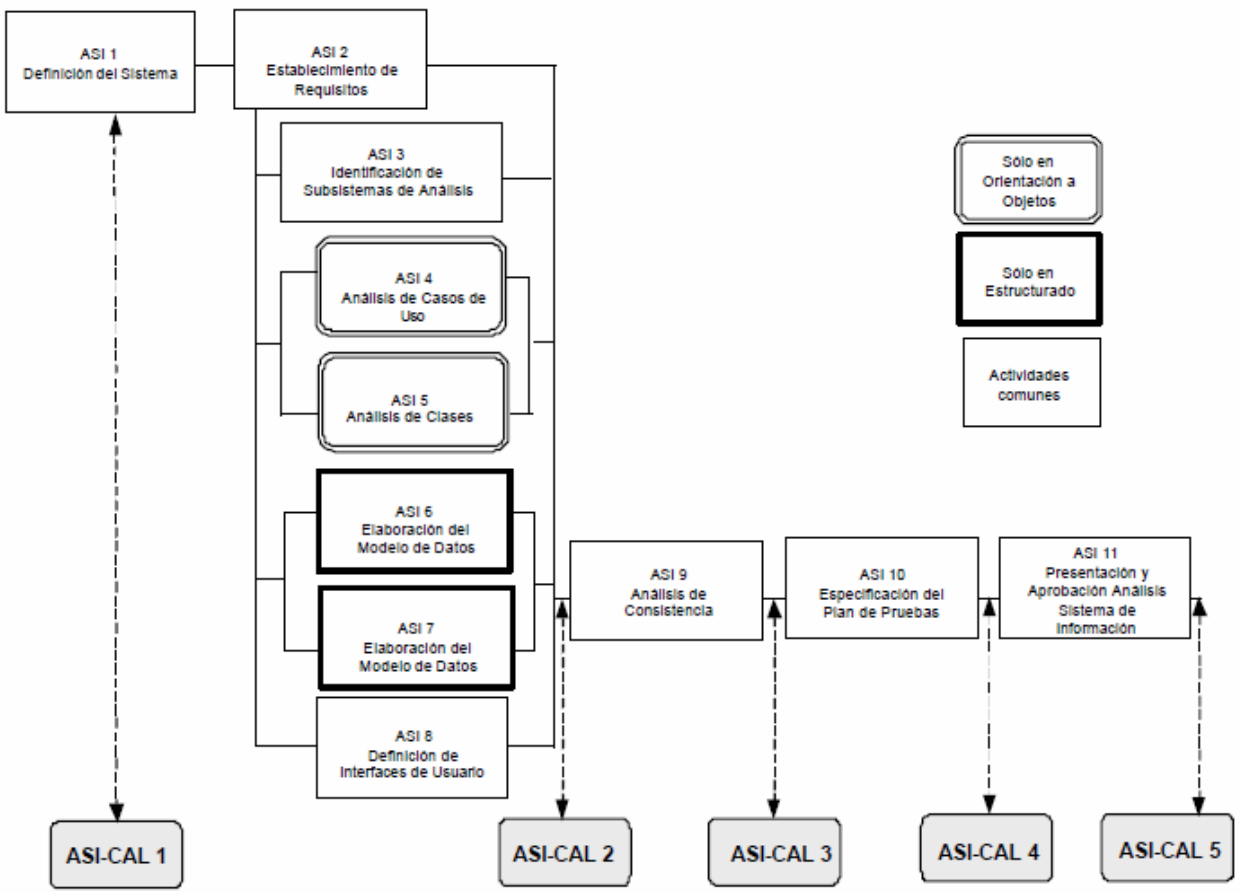


Ilustración 23: Fase ASI Métrica3

- Fase DSI:

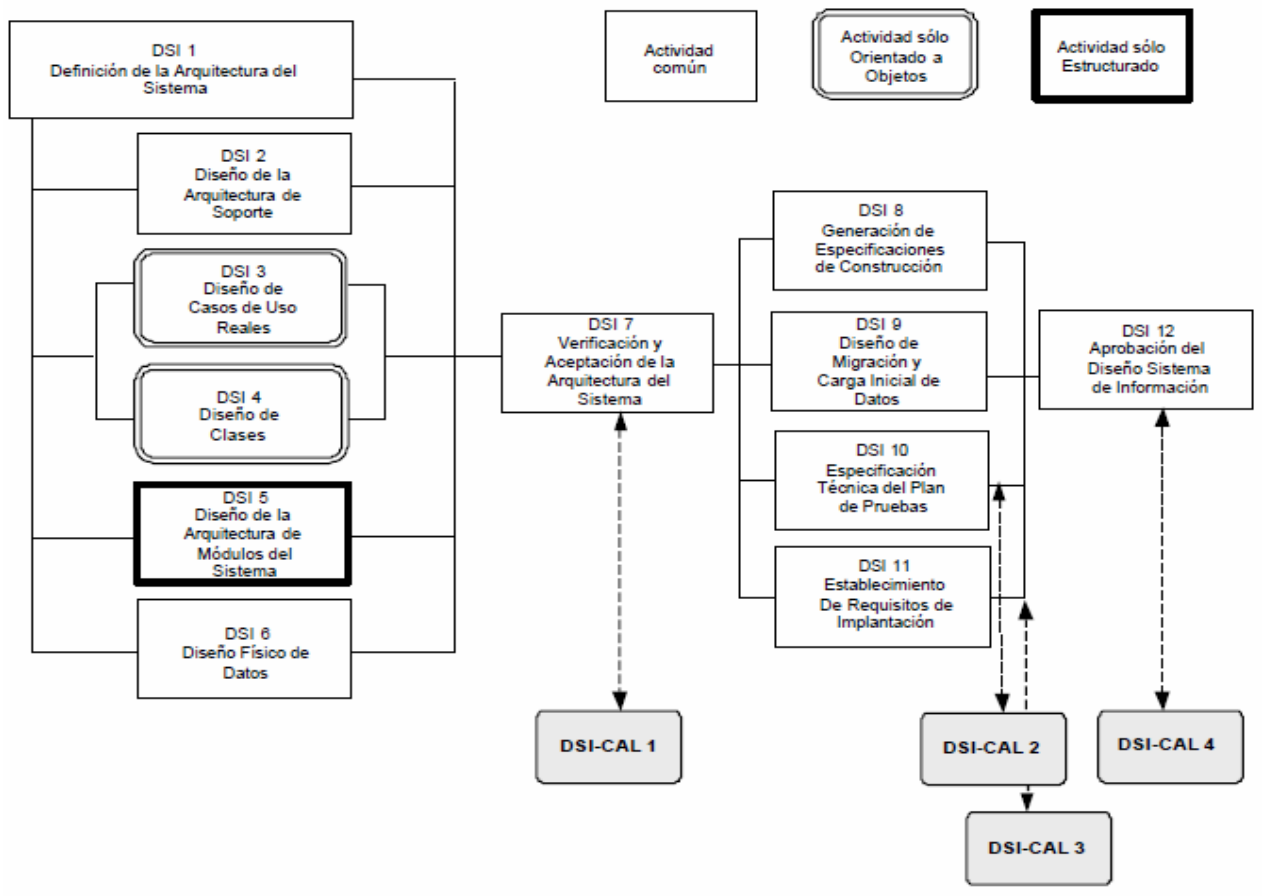


Ilustración 24: Fase DSI Métrica3

- Fase CSI:

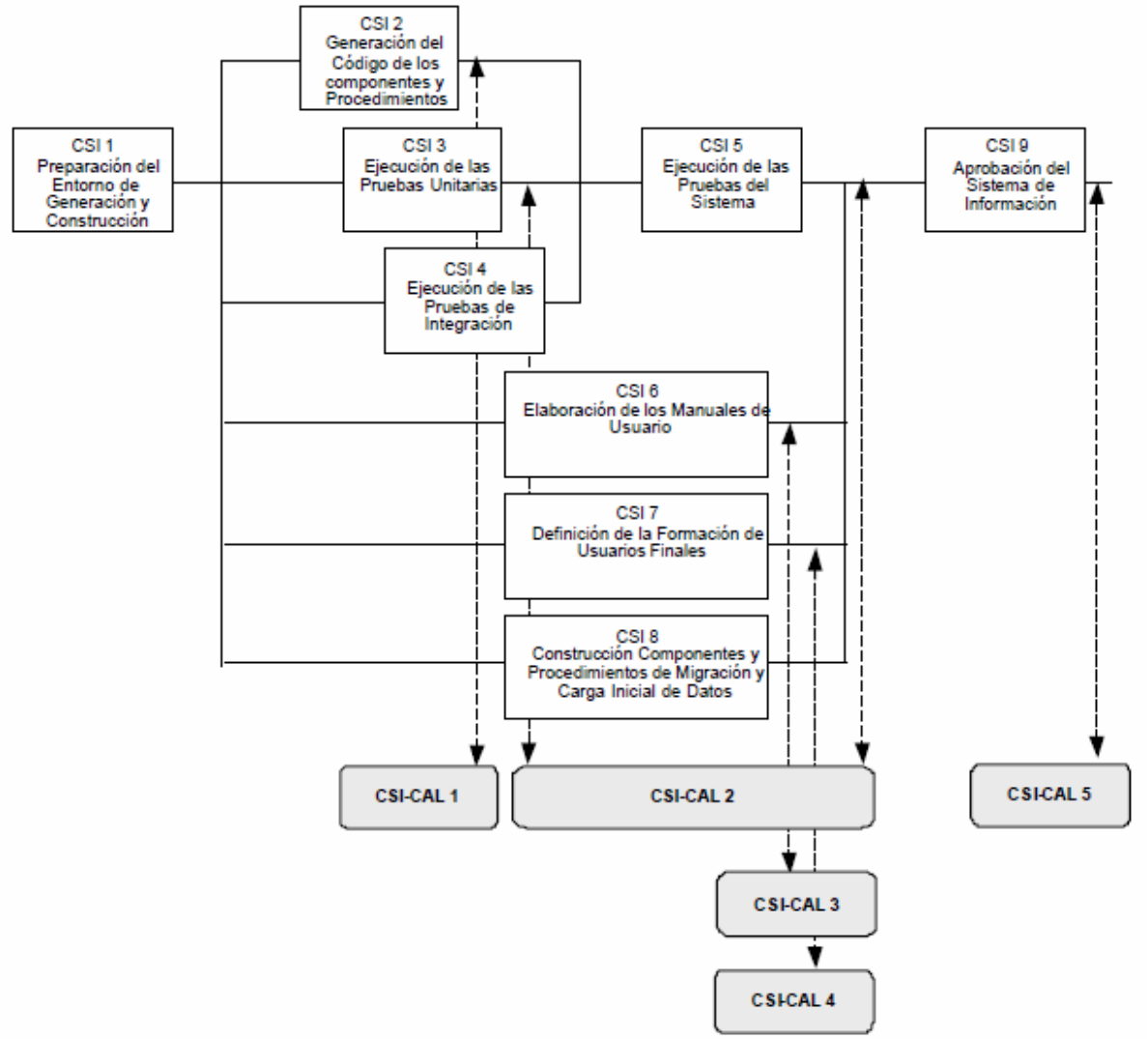


Ilustración 25: Fase CSI Métrica3

- Fase IAS:

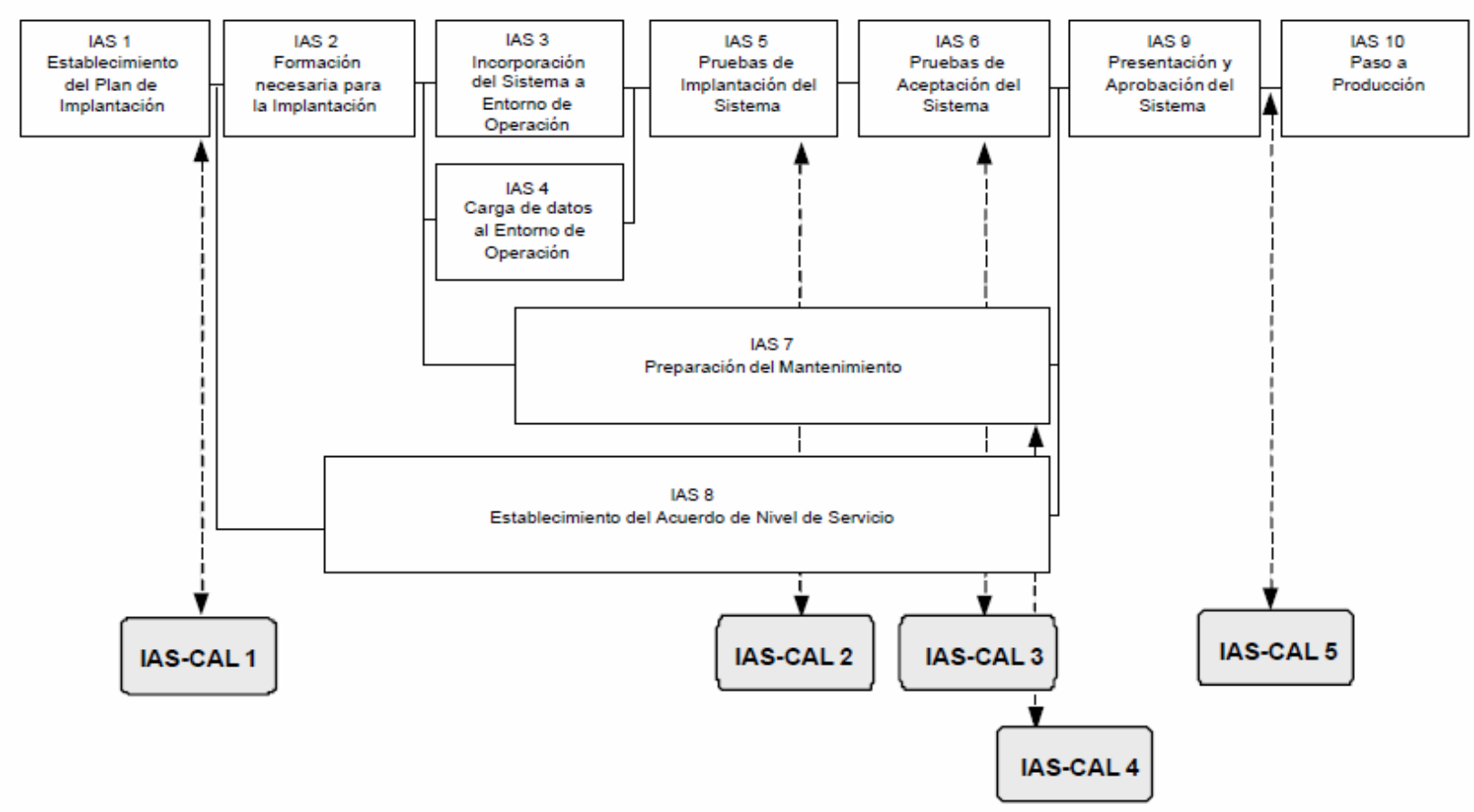


Ilustración 26: Fase IAS Métrica3

- Fase MSI:

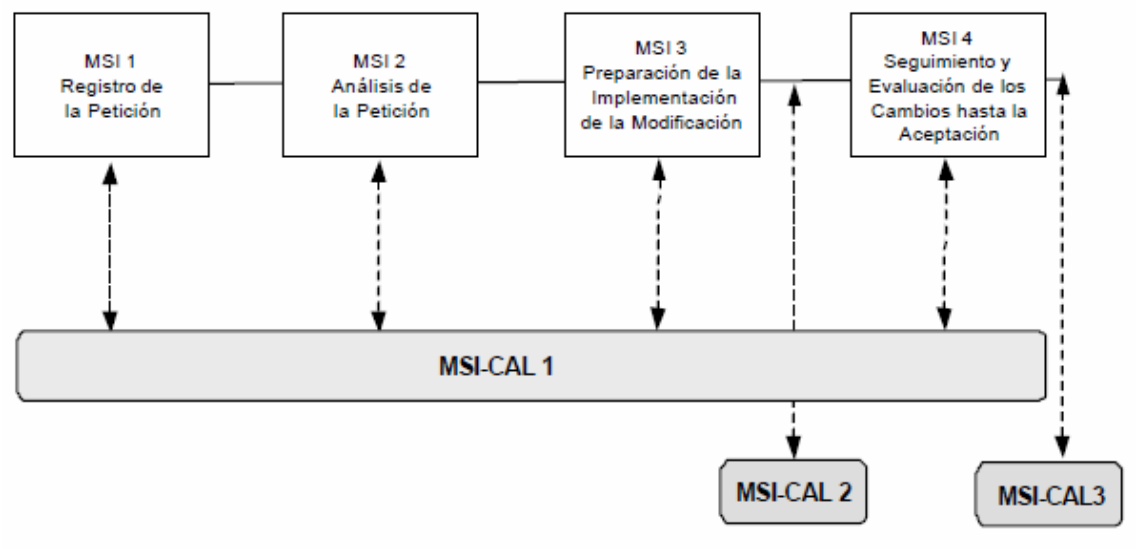


Ilustración 27: Fase MSI Métrica3

4.3.3. Conclusiones del caso

En este caso en el que se describe de la estructura principal de la metodología Métrica3, se puede observar que además de los procesos básicos para la creación de un determinado software se hace uso de una serie de metodologías adicionales. Estas se anexionan al proceso básico de desarrollo de aplicaciones mediante relaciones entre actividades. Una de ellas, la más importante respecto al punto de vista de este documento, es la de Calidad.

Este hecho mencionado indica que en base a un proceso establecido, pueden incorporarse nuevas prácticas y procesos al mismo con motivo de complementarlo.

III

Metodología de calidad sobre UI de aplicaciones WPF

III. Metodología de calidad sobre UI de aplicaciones WPF

1. Objetivos

El objetivo de este apartado es el de facilitar el aseguramiento de calidad en la utilización de aplicaciones visuales de escritorio programadas en lenguaje WPF (Windows Presentation Foundation). Para ello se proporcionará un marco común de referencia a la hora de realizar dicho proceso.

Hay que mencionar que se trata de un documento genérico en el que poder apoyarse, pero que para cada proyecto podrá variar en función a las características del mismo y a las condiciones dadas.

Con dicho plan de aseguramiento de la calidad se pretende *reducir*, *eliminar* y lo más importante, *prevenir* las deficiencias de calidad de los productos a obtener. Con ello se pretende obtener la confianza necesaria que garantice que las prestaciones y servicios esperados por el cliente o el usuario queden satisfechas.

Cabe destacar que la creación de la metodología está enmarcada en un proyecto de fin de carrera, por lo que se considera oportuno introducir diferentes anotaciones didácticas en el lugar en el que se precise, con motivo de aumentar el entendimiento de la misma.

1.1. Presentación del documento

El documento de la metodología desarrollada se compone de los siguientes apartados:

- Gestión de la configuración.
- Matrices resumen de la metodología.
- Descripción de la metodología.

En este último punto, el de descripción de la metodología, es donde se expondrá el cuerpo de la misma. Su estructura estará compuesta por:

- Fases: en cada una de ellas se hará una breve descripción de las necesidades que se plantean así como los objetivos que se pretenden alcanzar en cada una de las mismas.

- Actividades: cada fase de las anteriores mencionadas se compondrá de *actividades*. Por cada actividad se mostrará la descripción y objetivos que se pretenden obtener, pero esta vez de una forma más clara y concisa, haciendo uso de tablas. A continuación se muestra la plantilla que se utilizará en el documento para las actividades:

Descripción de actividades
Descripción
Objetivos

Tabla 1: Descripción de actividades

- Tareas: las actividades se desglosarán en *tareas*. En las mismas se mostrará una descripción de los pasos a seguir, los objetivos a alcanzar, los roles implicados, las entradas que se utilizan para la posterior creación del producto, así como el propio producto generado en dicha tarea. A continuación se muestra la plantilla que se utilizará en el documento para las tareas:

Descripción de tareas
Roles implicados
Descripción
Objetivos
Entradas
Producto generado

Tabla 2: Descripción de tareas

En resumen, el diagrama que relaciona el concepto de fase, actividad y tarea, contenidos en el apartado *descripción de la metodología*, quedaría como sigue:

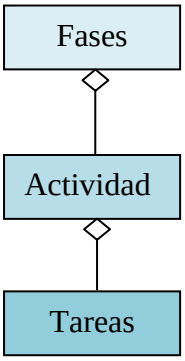


Ilustración 28: Desglose metodología

Normalmente cada actividad tendrá un marco temporal asociado dentro de cada fase. Con esto se pretende aclarar que cada una de ellas deberá ser realizada en un determinado orden, el cual se expondrá al comienzo de cada fase en forma de diagrama. Lo mismo ocurrirá con las tareas de cada actividad.

A continuación se muestra una posible distribución del diagrama temporal de actividades o tareas:

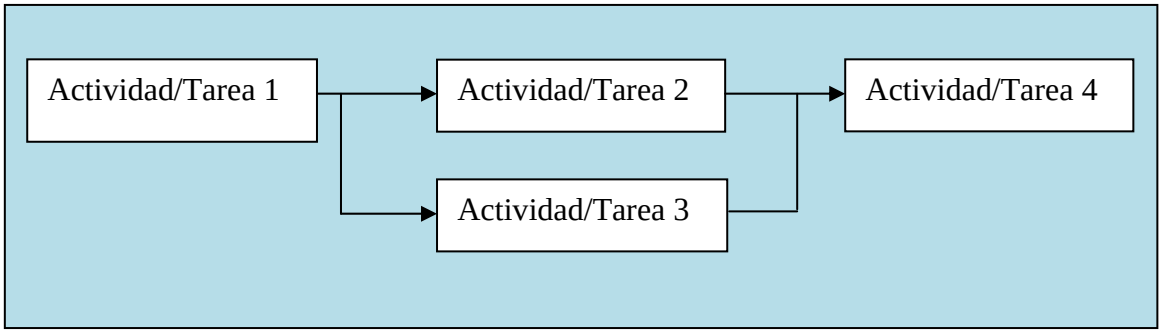


Ilustración 29: Diagrama temporal de actividades/tareas

Como se puede apreciar, para poder llevar a cabo la actividad o tarea 2 y 3 se deberá realizar la actividad o tarea 1. En caso de la segunda y la tercera podrán ser realizadas de forma paralela, y habrá que terminar ambas para comenzar la actividad o tarea número 4.

1.2. Gestión de la configuración

Cada producto que se genere en las sucesivas tareas de la metodología deberá tener el nombre especificado según corresponda, almacenando cada uno de ellos en un dispositivo de almacenamiento con algún tipo de estructura jerárquica que refleje en qué fase, actividad y tarea han sido creados. Un ejemplo de lo mencionado puede ser el siguiente espacio de directorios expuesto, dentro de un dispositivo de almacenamiento no volátil:

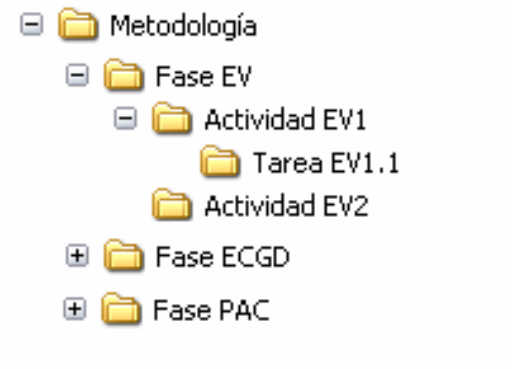


Ilustración 30: Gestión de la configuración

1.3. Matrices resumen de la metodología

A continuación se muestran una serie de matrices resumen de la metodología que pueden ser de utilidad.

1.3.1. Matriz de tareas/roles:

En estas matrices se muestra la relación directa entre los diferentes roles que según la metodología deben existir (columnas), y las diferentes tareas dentro de cada fase (filas). Por relación se entiende, colaboración de un determinado rol en cierta tarea.

1.3.1.1. Matriz EVS

Tarea/Rol	JEC	Analista de sistemas	Analista de pruebas	Tester	Representante del equipo de desarrollo
EV1.1		X			
EV2.1	X				
EV2.2	X				
EV2.3	X	X			

Tabla 3: matriz tarea/rol EVS

1.3.1.2. Matriz ECGD

Tarea/Rol	JEC	Analista de sistemas	Analista de pruebas	Tester	Representante del equipo de desarrollo
ECGD1.1		X			
ECGD1.2	X		X		X
ECGD2.1	X		X		
ECGD2.2			X	X	
ECGD2.3		X			

Tabla 4: matriz tarea/rol ECGD

1.3.1.3. Matriz PAC

Tarea/Rol	JEC	Analista de sistemas	Analista de pruebas	Tester	Representante del equipo de desarrollo
PAC1.1	X		X		
PAC1.2	X		X		
PAC1.3			X	X	
PAC2.1		X		X	
PAC2.2				X	
PAC3.1				X	

PAC3.2	X			X	X

Tabla 5: matriz tarea/rol PAC

1.3.2. Matriz de tareas/productos:

En estas matrices se muestra la relación directa entre los diferentes productos generados (columnas), y las diferentes tareas dentro de cada fase (filas). Por relación se entiende, en qué tarea ha sido desarrollado un determinado producto.

1.3.2.1. Matriz EVS

Tarea/Producto	REQ-CARAC-SIST	EQUIP-CAL	EQUIP-CAL-SUPL	Aprob-Viabilidad
EV1.1	X			
EV2.1	X	X		
EV2.2			X	
EV2.3				X

Tabla 6: matriz tarea/producto EVS

1.3.2.2. Matriz ECGD

Tarea/Producto	Módulos-Funcionales	Listado-Casos-Uso	Casos-Uso-Candidatos	Casos-Uso-Objeto	Controles-Casos-Uso	Detalle-Controles
ECGD1.1	X					
ECGD 1.2	X	X	X			
ECGD 2.1				X		
ECGD 2.2					X	
ECGD 2.3						X

Tabla 7: matriz tarea/producto ECGD

1.3.2.3. Matriz PAC

Tarea/Producto	Planes-Prueba	Casos-Prueba	Scripts-Caso-Prueba	Entorno-Pruebas	Resultado-Casos-Prueba	Listado-Bugs
PAC1.1	**X**					
PAC 1.2		**X**				
PAC 1.3			**X**			
PAC 2.1				**X**		
PAC 2.2					**X**	
PAC 3.1					**X**	
PAC 3.2						**X**

Tabla 8: matriz tarea/producto PAC

2. Descripción de la metodología

A continuación se presenta el documento de Metodología de calidad para el aseguramiento de calidad en aplicaciones visuales WPF.

EV - Estudio de Viabilidad

En primer lugar, se debe realizar un estudio para comprobar la viabilidad del proyecto. En este caso se trata de un proyecto de calidad, por lo que dicha fase será enfocada como tal, es decir, se deberá verificar si el producto del que se va a garantizar la calidad es compatible con los requisitos que enuncia la metodología.

En este estudio se deberá disponer de un rol inicial al cual se denominará *analista de sistemas.* Este rol será el encargado de llevar a cabo el análisis de viabilidad, por lo que dependiendo de la decisión final que tome, se continuará con el proyecto, o se considerará que no es factible llevarlo a cabo.

En dicho estudio, y en caso de que los requisitos de viabilidad necesarios sean completamente satisfechos, se deberá crear un equipo de calidad que pueda trabajar de forma adecuada acorde a las características del sistema del que se pretende garantizar la calidad. En caso contrario, el aseguramiento de calidad no será viable.

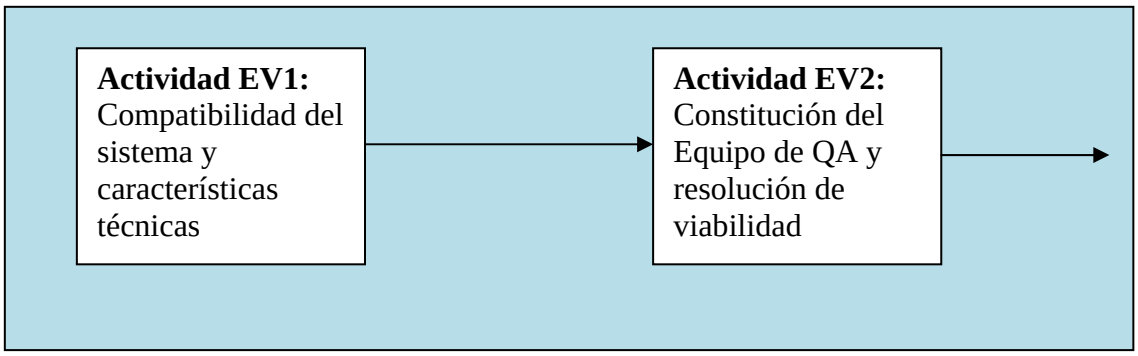


Ilustración 31: Diagrama temporal de actividades EV

Actividad EV1 - Compatibilidad del sistema y características técnicas

Actividad EV1	
Descripción	Verificar si la aplicación sobre la que se pretende garantizar la calidad es compatible con la metodología. Se deben extraer datos importantes sobre cómo está desarrollada la aplicación y qué tecnologías utiliza para conocer si se cuenta con los recursos humanos y materiales necesarios para llevar a cabo el plan de aseguramiento de calidad.
Objetivos	Conocer los detalles de la aplicación en términos de características técnicas y tecnologías que emplea.

Tabla 9: Actividad EV1

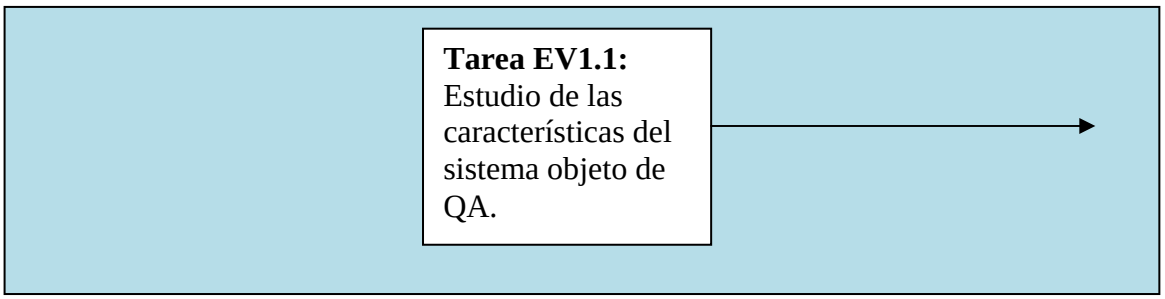


Ilustración 32: Diagrama temporal de tareas EV1

Tarea EV1.1: Estudio de las características del sistema objeto de QA

Tarea EV1.1	
Roles implicados	Analista de sistemas
Descripción	Estudio previo sobre la aplicación objeto de aseguramiento de calidad, así como de la documentación pertinente en caso de existir, para garantizar que se cumplen los requisitos técnicos necesarios para ser plenamente compatible con la metodología. Dichos requisitos *necesarios* son los siguientes: ▪ Aplicación visual. ▪ Aplicación WPF (Windows Presentation Foundation) de Microsoft. ▪ Acceso permitido a la UI mediante programación. ▪ Capacidad de referenciar unívocamente a todos los elementos de la interfaz. Se debe extraer información sobre lo siguiente, para valorarlo en posteriores tareas: ▪ Características propias de la aplicación sobre la que se garantiza la calidad. Estas pueden ser del tipo: o Lenguaje de programación en el que está implementada la aplicación (C#, VB…) o Framework en el que ha sido desarrollada la aplicación (3.0 ó 3.5). o Si requiere acceso a bases de datos, en caso afirmativo qué tecnología emplea.

	○ Otras tecnologías externas que utilice la aplicación. ▪ Verificar que los sistemas operativos de las máquinas donde se van a realizar las pruebas son Windows XP o Windows Vista y cumplen con los requisitos implícitos de las características propias de la aplicación.
Objetivos	Obtención de una lista de características propias del sistema a garantizar la calidad. Entre ellas deberán encontrarse las necesarias para garantizar la viabilidad. Con esto último se conseguirá información muy importante relativa a la viabilidad del proyecto ya que dadas determinadas características del sistema pueden hacer que el equipo de calidad no esté preparado para garantizar la calidad del mismo, por falta de experiencia, conocimientos, o simplemente por no disponer de los recursos o herramientas necesarias para tal fin.
Entradas	▪ Documentos de análisis y diseño de la aplicación. ▪ Código de la aplicación.
Producto generado	▪ Req-Carac-Sist: Documento que contendrá las características del sistema extraídas durante la actual tarea. Por cada característica mencionada, se deberá mostrar lo siguiente: ○ Descripción concisa de la característica. ○ Campo de observaciones, el cual deberá ser cumplimentado en caso de existir alguna observación (positiva o negativa).

	o Campo en el que se marcará si la característica es favorable o no, dadas las observaciones en caso de que exista alguna. Este último campo estará sujeto a modificaciones ya que aún no se sabe si posteriormente se añadirán nuevas observaciones a las propias características que hagan que el campo de *favorable* pueda variar.

Tabla 10: Tarea EV1.1

Plantilla correspondiente al producto de salida Req-Carac-Sist de la tarea EV1.1:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Req-Carac-Sist

REQ-CARAC-SIST		
CUMPLIMIENTO REQUISITOS NECESARIOS		
Aplicación WPF	Si ☐	No ☐
Aplicación Visual	Si ☐	No ☐
Acceso permitido a la UI mediante programación	Si ☐	No ☐
Controles identificables	Si ☐	No ☐
Máquinas con S.O. Windows Xp o Vista	Si ☐	No ☐
ESTUDIO DE CARACTERÍSTICAS DEL SISTEMA		
Descripción		
Observaciones		
Favorable	Si ☐	No ☐
...		

Plantilla 1: Req-Carac-Sist

Actividad EV2 - Constitución del Equipo de QA y resolución de viabilidad

Actividad EV2	
Descripción	Formar un equipo de calidad, el cual será el encargado de verificar la calidad de la aplicación. El jefe de calidad será el encargado de formar el resto del equipo. Los roles que deben existir obligatoriamente en un proyecto de este tipo se enumeran a continuación, definiendo las responsabilidades asociadas a cada uno de los mismos en términos de la metodología:

Jefe encargado de Calidad:
Tarea EV2.1: Asignación de personal a los roles identificados. Asignar el personal pertinente a cada rol identificado.
Tarea EV2.2: Asignación de personal suplente a los roles identificados. Asignar el personal suplente pertinente a cada rol identificado.
Tarea EV2.3: Aceptación o rechazo del estudio de viabilidad. Tomar la decisión final sobre la viabilidad del proyecto.
Tarea ECGD1.2: Obtención de casos de uso candidatos. Verificar el listado de casos de uso definidos y remitírselos al responsable del equipo de desarrollo en caso de estar de acuerdo.
Tarea ECGD2.1: Casos de uso objeto de QA.

Dar la conformidad respecto a los casos de uso objeto de QA que se han establecido.
Tarea PAC1.1: Descripción de Planes de Prueba. ▪ Estudiar la documentación generada durante la fase ECGD. ▪ Crear los planes de prueba.
Tarea PAC1.2: Definición de casos de Prueba. Crear los casos de prueba.
Tarea PAC3.2: Notificación de defectos al equipo de desarrollo. Revisión y posteriormente reporte del listado de defectos al representante del equipo de desarrollo.

Analista de sistemas:
Tarea EV1.1: Estudio de las características del sistema objeto de QA. Estudio de la aplicación y la documentación pertinente generada.
Tarea EV2.3: Aceptación o rechazo del estudio de viabilidad. Determinar si el proyecto es viable.
Tarea ECGD1.1: Extracción de módulos funcionales. Estudiar los diferentes módulos funcionales de la aplicación y sus respectivas funcionalidades.
Tarea ECGD2.3: Estudio de propiedades y eventos asociados a los controles objeto de QA.

Estudiar las propiedades más relevantes de los controles identificados en la anterior tarea.
Tarea PAC2.1: Preparación para la ejecución de casos de prueba. Estudiar el entorno que sería necesario para simular el del cliente.

Analista de pruebas:
Tarea ECGD1.2: Obtención de casos de uso candidatos. ▪ Estudiar los casos de uso existentes en la documentación del sistema. ▪ Encontrar otros posibles casos de uso de interés
Tarea ECGD2.1: Casos de uso objeto de QA. Estudiar qué casos de uso van a ser objeto de QA.
Tarea ECGD2.2: Controles por caso de uso. Supervisar al Tester en la tarea de determinar qué controles intervienen para llevar a cabo cada uno de los casos de uso.
Tarea PAC1.1: Descripción de Planes de Prueba. ▪ Estudiar la documentación generada durante la fase ECGD. ▪ Crear los planes de prueba.
Tarea PAC1.2: Definición de casos de Prueba. Crear los casos de prueba.
Tarea PAC1.3: Creación de scripts de Prueba. Analizar si es viable automatizar un determinado caso de prueba.

Tester:
Tarea ECGD2.2: Controles por caso de uso. Determinar qué controles intervienen para llevar a cabo cada uno de los casos de uso.
Tarea PAC1.3: Creación de scripts de Prueba. Generar los scripts correspondientes a los casos de prueba.
Tarea PAC2.1: Preparación para la ejecución de casos de prueba Preparar el entorno necesario establecido por el analista de sistemas.
Tarea PAC2.2: Puesta en marcha de los casos de prueba. Ejecución de los casos de prueba.
Tarea PAC3.1: Revisión de casos de Prueba. Verificar los resultados obtenidos en los casos de prueba.
Tarea PAC3.2: Notificación de defectos al equipo de desarrollo. Crear un listado con los defectos encontrados.

Representante del equipo de desarrollo:
Tarea ECGD1.2: Obtención de casos de uso candidatos. Recoger el listado de casos de uso y remitírselos al equipo de desarrollo.

	Tarea PAC3.2: Notificación de defectos al equipo de desarrollo. Recoger el listado de defectos para estudiarlos.
Objetivos	Determinación del equipo de calidad, asociando personal a cada rol identificado anteriormente.

Tabla 11: Actividad EV2

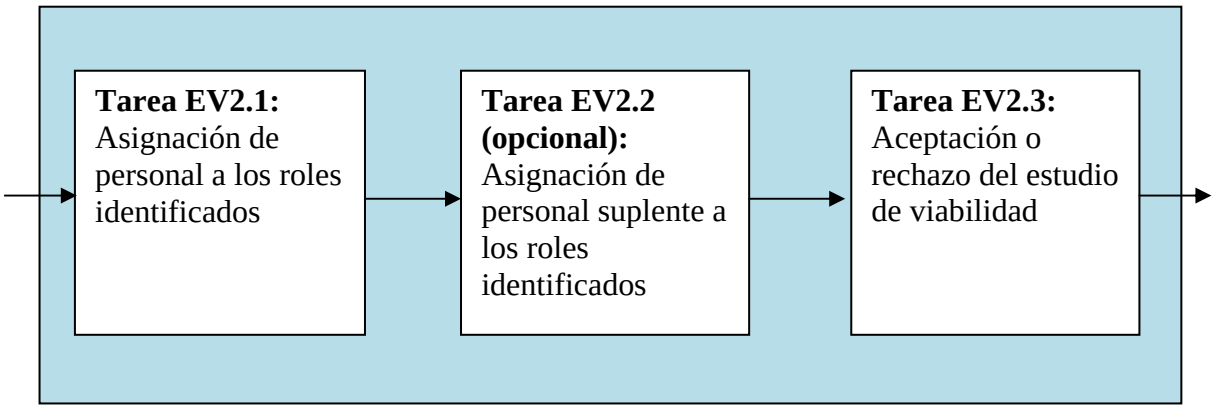


Ilustración 33: Diagrama temporal de tareas EV2

Tarea EV2.1: Asignación de personal a los roles identificados

Tarea EV2.1	
Roles implicados	JEC
Descripción	Asignar el personal pertinente a cada rol identificado anteriormente por parte del JEC. La elección deberá estar fundamentada de tal forma que las capacidades de la persona a asignar sean acordes con las responsabilidades que conlleva

	desempeñar el propio rol especificado. Dependiendo de la envergadura del proyecto de calidad, podrá haber diferente número de personas asignadas a un mismo rol, quedando todas ellas dependiendo de su padre en la jerarquía. Convenientemente la elección del personal, debería tener relación con el producto *Req-Carac-Sist* generado en anteriores actividades. Esto es debido a que en dicho documento se exponen las principales características del sistema, por lo que si el personal de que se dispone es compatible a dichas características el rendimiento, y por lo tanto la viabilidad del proyecto será mayor.
Objetivos	Obtención de los recursos humanos que formarán el equipo de calidad, identificados cada uno de ellos por un determinado rol.
Entradas	▪ Req-Carac-Sist
Producto generado	▪ Req-Carac-Sist modificado (opcionalmente): Puede ser necesario modificar el campo *observaciones* o *favorable* del documento mencionado (REQ-CARAC-SIST) ya que en función al equipo de calidad elegido puede haber alguna *observación* positiva o negativa que haga que el campo *favorable* pueda variar. ▪ Equip-Cal: o Informe en el que se constate qué personas están asociadas a los diferentes roles.

	o Motivo por el que se ha elegido a una determinada persona. o Opcionalmente se podrá crear un diagrama en el que se pueda plasmar la jerarquía relativa a los roles involucrados en el proceso, especificando las personas asociadas a cada rol.

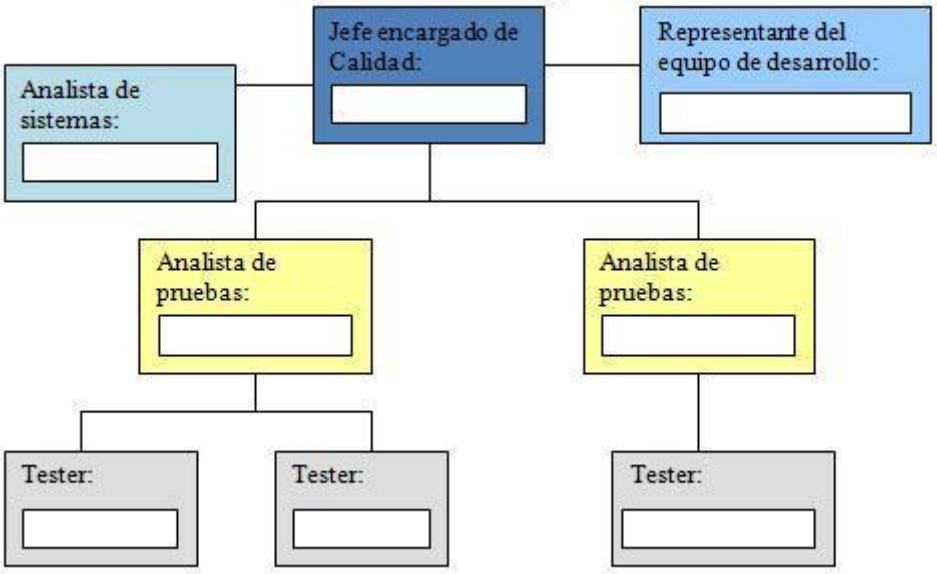
Tabla 12: Tarea EV2.1

Plantilla correspondiente al producto de salida Equip-Cal de la tarea EV2.1:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Equip-Cal

EQUIP-CAL	
Jefe encargado de Calidad	
Motivo de la elección	
Representante del equipo de desarrollo	
Motivo de la elección	
Analista de sistemas	
Motivo de la elección	
Analista de pruebas	
Motivo de la elección	
Tester	
Motivo de la elección	

Plantilla 2: Equip-Cal 1



Plantilla 3: Equip-Cal 2

Tarea EV2.2: Asignación de personal suplente a los roles identificados (opcional)

Tarea EV2.2	
Roles implicados	JEC
Descripción	Asignar al menos una persona por rol, la cual pudiera desempeñar sus funciones normalmente en caso de ser necesario.
Objetivos	Tener constancia de qué personas pueden suplir a otras pertenecientes a un determinado rol.
Entradas	▪ Req-Carac-Sist
Producto generado	▪ Equip-Cal-Supl: Tendrá la misma plantilla que *Equip-Cal*, sin exponer la jerarquía entre roles.

Tabla 13: Tarea EV2.2

Plantilla correspondiente al producto de salida Equip-Cal-Supl de la tarea EV2.2:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Equip-Cal-Supl

EQUIP-CAL-SUPL	
Jefe encargado de Calidad	
Motivo de la elección	
Representante del equipo de desarrollo	
Motivo de la elección	
Analista de sistemas	
Motivo de la elección	
Analista de pruebas	
Motivo de la elección	
Tester	
Motivo de la elección	

Plantilla 4: Equip-Cal-Supl

Tarea EV2.3: Aceptación o rechazo del estudio de viabilidad

Tarea EV2.3	
Roles implicados	JEC Analista de sistemas
Descripción	Determinar por parte del analista de sistemas en compañía del JEC si el proyecto es viable. Su decisión será tomada en función al documento *Req-Carac-Sist*, verificando las características favorables y desfavorables del correspondiente documento para emitir el veredicto final.
Objetivos	Conocer la resolución de viabilidad del proyecto para saber si se seguirá con el mismo.
Entradas	▪ Req-Carac-Sist
Producto generado	▪ Aprob-Viabilidad: Informe en el que se constate la viabilidad del proyecto. Dicho documento deberá contener lo siguiente: o Observaciones relativas a la viabilidad del proyecto. En este apartado se debe explicar la problemática de las características catalogadas como desfavorables en el documento *Req-Carac-Sist*. o Razones por las que se ha aprobado/rechazado la viabilidad del proyecto. o Resolución de viabilidad: especificar si el proyecto

	es viable. o Firma del *Analista de sistemas* y del *Jefe encargado de calidad.*

Tabla 14: Tarea EV2.3

Plantilla correspondiente al producto de salida Aprob-Viabilidad de la tarea EV2.3:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Aprob-Viabilidad

Aprob-Viabilidad			
Observaciones pertinentes			
Observación:			
…			
Motivos por los que se aprueba/deniega la viabilidad del proyecto			
Aprobado ☐		**Denegado** ☐	
Analista de sistemas		Firma:	
Jefe encargado de calidad		Firma:	
	Fecha		

Plantilla 5: Aprob-Viabilidad

ECGD – Extracción de conocimiento y generación de documentación preliminar.

Anotación didáctica

Todo proyecto de aseguramiento de calidad sobre una determinada aplicación deberá comenzar con el estudio intensivo de la propia aplicación. Normalmente las personas que desarrollan la aplicación no son las mismas que van a poner en marcha el plan de aseguramiento de calidad por lo que no sabrán nada acerca de la misma. Por esto último mencionado esta fase es imprescindible para llegar a buen puerto.

El cometido de esta fase es el de obtener toda la documentación necesaria relativa a la aplicación en cuestión para poder llevar a cabo la creación de planes de pruebas de forma exitosa.

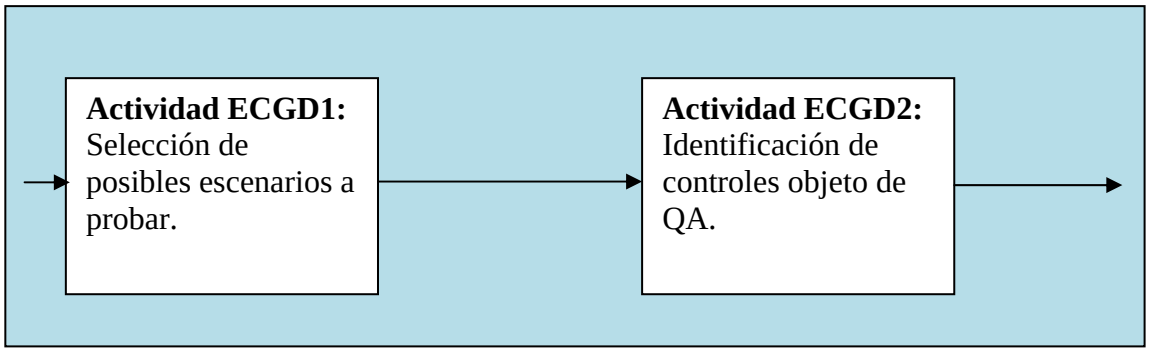


Ilustración 34: Diagrama temporal de actividades ECGD

Actividad ECGD1- Selección de posibles escenarios a probar

Actividad ECGD1	
Descripción	En esta actividad se deberán obtener diferentes casos de uso de la aplicación, los cuales deberán catalogarse en función al módulo funcional al que pertenezcan.

Objetivos	Proponer los casos de uso elegidos como posibles futuros escenarios de casos de prueba.

Tabla 15: Actividad ECGD1

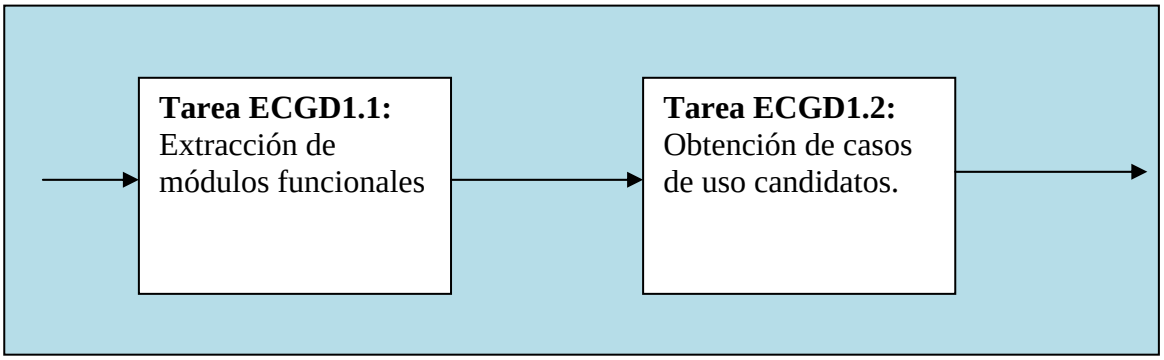


Ilustración 35: Diagrama temporal de tareas ECGD1

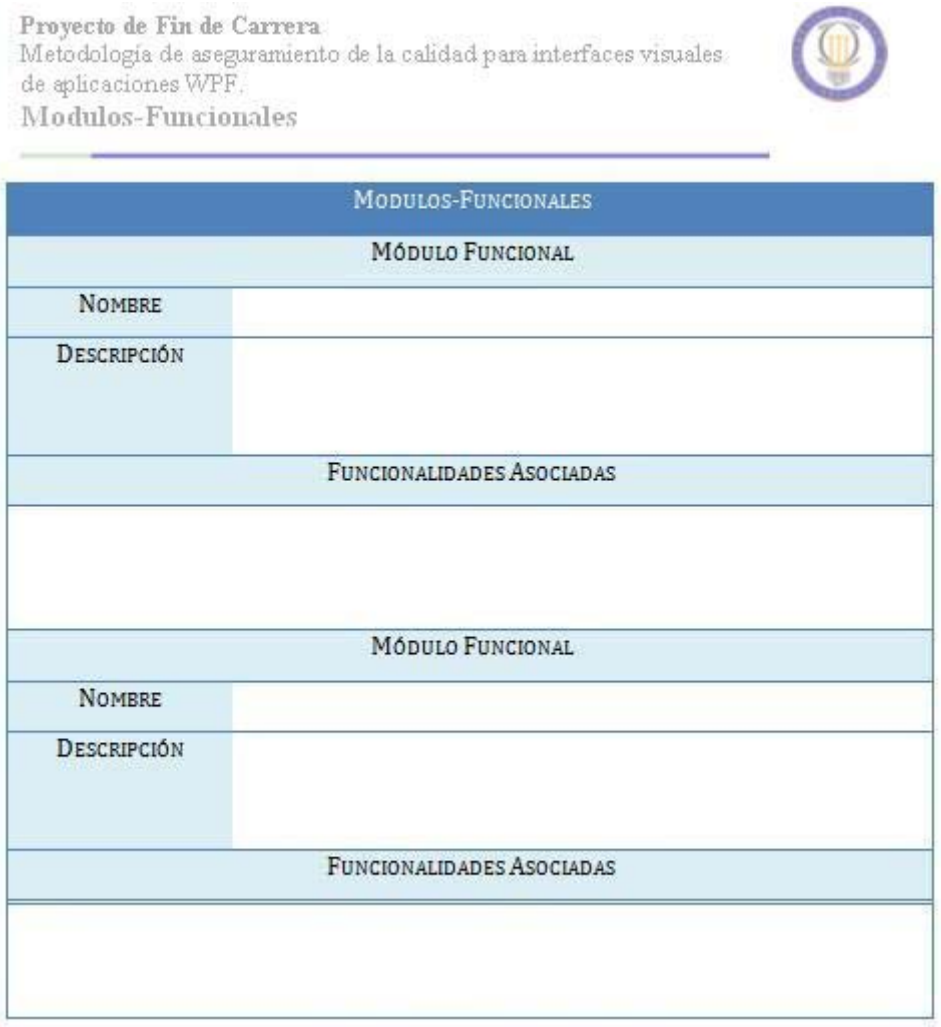
Tarea ECGD1.1: Extracción de módulos funcionales

Tarea ECGD1.1	
Roles implicados	Analista de sistemas
Descripción	Estudiar los diferentes módulos funcionales de la aplicación a través de la documentación pertinente o del ejercicio directo sobre la propia aplicación (a través de su ejecutable), los cuales contendrán un conjunto de funcionalidades. Describir las funcionalidades asociadas a cada módulo funcional.
Objetivos	Obtención de las diferentes funcionalidades de la aplicación.
Entradas	▪ Documentos de análisis y diseño de la aplicación.

	▪ Ejecutable de la aplicación.
Producto generado	▪ Módulos-Funcionales: Documento que detalle los módulos funcionales del sistema, especificando las funcionalidades asociadas a cada uno de ellos. Por cada módulo funcional se debe indicar: o Nombre del módulo: nombre descriptivo del módulo funcional. o Descripción del módulo: descripción de la funcionalidad contenida en el módulo funcional. o Descripción de las funcionalidades asociadas.

Tabla 16: Tarea ECGD1.1

Plantilla correspondiente al producto de salida Módulos-Funcionales de la tarea ECGD1.1:



Plantilla 6: Módulos-Funcionales

Tarea ECGD1.2: Obtención de casos de uso candidatos

Tarea ECGD1.2	
Roles implicados	JEC Analista de pruebas Representante del equipo de desarrollo

Descripción	Estudiar por parte del analista de pruebas los casos de uso existentes en la documentación del sistema en caso de existir y encontrar otros de interés, ayudándose del documento *Módulos-Funcionales*. Completar el documento *Módulos-Funcionales* si se considera oportuno. Hacer un listado de casos de uso que deberían estar definidos y no lo están, remitiendo posteriormente dicho listado por parte del JEC al responsable del equipo de desarrollo. Establecer una correspondencia entre los casos de uso que se considere oportuno y los módulos funcionales definidos con anterioridad. El JEC deberá dar su conformidad respecto a los casos de uso candidatos que se han establecido.
Objetivos	Disponer de todos los casos de uso necesarios.
Entradas	▪ Casos de uso creados por el equipo de desarrollo (si existen). ▪ Módulos-Funcionales
Producto generado	▪ Módulos-Funcionales modificados (opcionalmente): Como consecuencia del estudio de los posibles casos de uso existentes en la documentación, puede surgir la modificación de mencionado documento.

- Listado-Casos-Uso:

Documento que detalle los casos de uso que deberían estar definidos y no lo están. Los campos a cumplimentar por cada caso de uso serán los siguientes:

  - Nombre del caso de uso.
  - Descripción del caso de uso: breve descripción de la funcionalidad que lleva asociada.

- Casos-Uso-Candidatos:

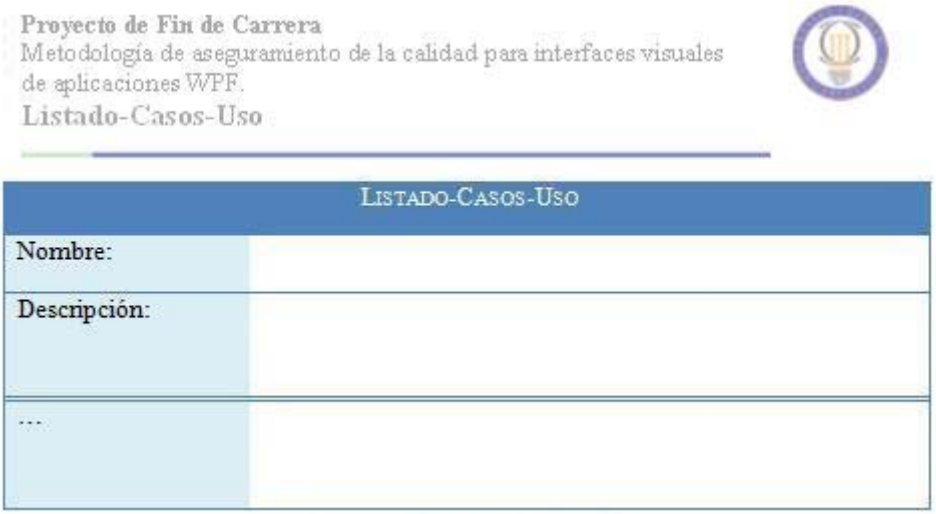
Documento que especifique todos y cada uno de los casos de uso que se pueden necesitar para las posteriores pruebas. Además se deberán relacionar con las funcionalidades extraídas del documento *Módulos-Funcionales*.

Cada entrada en el documento deberá contener lo siguiente:

  - Nombre de la funcionalidad según el documento *Módulos-Funcionales*.
  - Identificador del caso de uso en función al documento de casos de uso creado por el equipo de desarrollo.

Tabla 17: Tarea ECGD1.2

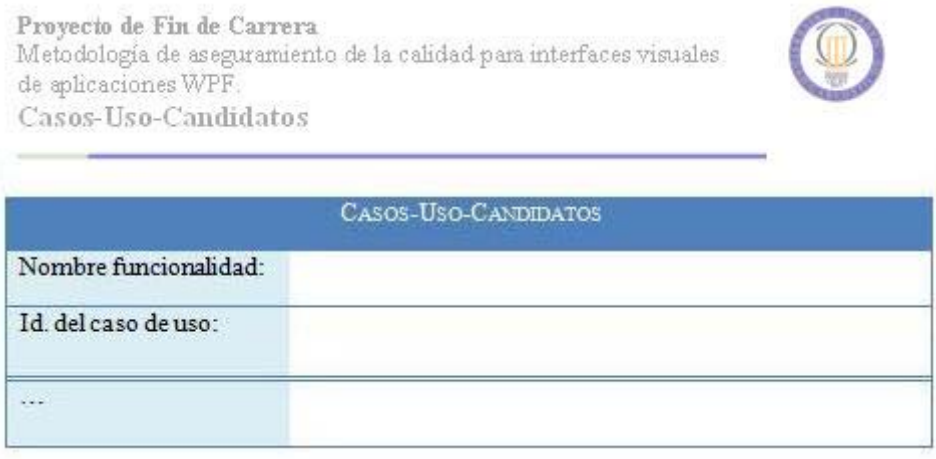
Plantilla correspondiente al producto de salida Listado-Casos-Uso de la tarea ECGD1.2:



LISTADO-CASOS-USO	
Nombre:	
Descripción:	
…	

Plantilla 7: Listado-Casos-Uso

Plantilla correspondiente al producto de salida Casos-Uso-Candidatos de la tarea ECGD1.2:



CASOS-USO-CANDIDATOS	
Nombre funcionalidad:	
Id. del caso de uso:	
…	

Plantilla 8: Casos-Uso-Candidatos

Actividad ECGD2- Identificación de controles objeto de QA

Actividad ECGD2	
Descripción	Identificar qué casos de uso de los extraídos en anteriores actividades van a ser objeto de QA. Además se deberá especificar qué controles están asociados a cada uno de ellos, estableciendo las características que tienen asociadas.
Objetivos	Recopilar toda la información necesaria por cada caso de uso para poder llevar a cabo las pruebas.

Tabla 18: Actividad ECGD2

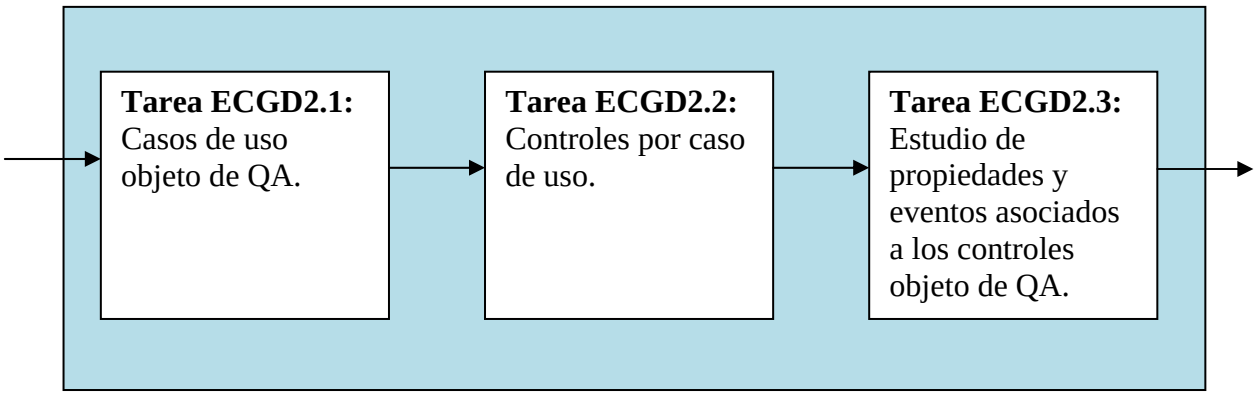


Ilustración 36: Diagrama temporal de tareas ECGD2

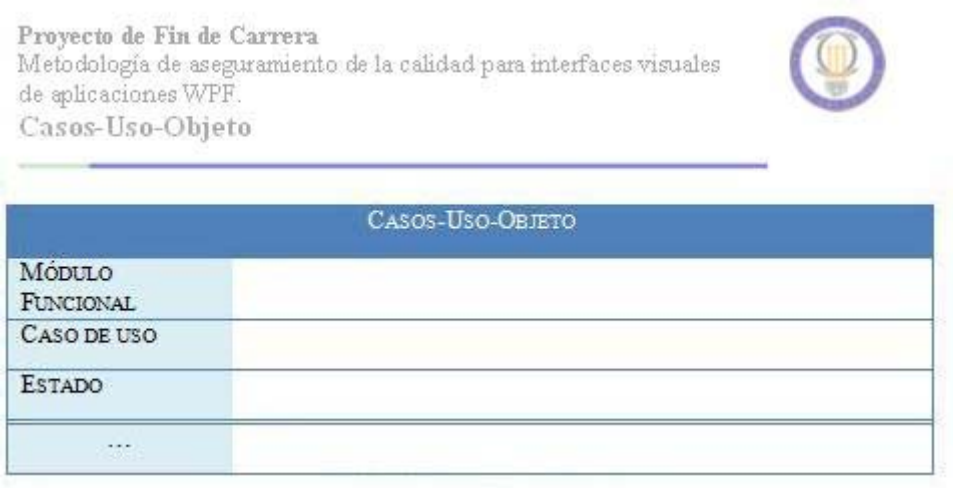
Tarea ECGD2.1: Casos de uso objeto de QA

Tarea ECGD2.1	
Roles implicados	JEC Analista de pruebas
Descripción	Determinar por parte del analista de pruebas qué casos de uso descritos en el documento *Casos-Uso-Candidatos* van a ser objeto de QA. Con ello, se tendrá conocimiento sobre qué cosas se pretenden probar, así como de verificar las que ya se han probado. El JEC deberá dar su conformidad respecto a los casos de uso objeto de QA que se han establecido.
Objetivos	Selección de los caso de uso que van a ser objeto de pruebas.
Entradas	▪ Casos-Uso-Candidatos ▪ Módulos-Funcionales
Producto generado	▪ Casos-Uso-Objeto: Documento que contendrá los casos de uso objeto de aseguramiento de calidad. Además, se expondrá el estado en el que se encuentra la ejecución del plan de calidad para cada caso de uso. Cada caso de uso contenido en el documento deberá contener los siguientes campos: o Módulo funcional: nombre del módulo funcional al

	que pertenece. o Caso de uso: nombre del caso de uso. o Estado: estado en el que se encuentra la verificación del cumplimiento de la funcionalidad asociada al caso de uso. Los posibles estados son: *parado*, *en proceso* o *finalizado.*

Tabla 19: Tarea ECGD2.1

Plantilla correspondiente al producto de salida Casos-Uso-Objeto de la tarea ECGD2.1:


Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Casos-Uso-Objeto

CASOS-USO-OBJETO	
MÓDULO FUNCIONAL	
CASO DE USO	
ESTADO	
...	

Plantilla 9: Casos-Uso-Objeto

Tarea ECGD2.2: Controles por caso de uso

Tarea ECGD2.2	
Roles implicados	Analista de pruebas Tester
Descripción	Determinar por parte del tester qué controles intervienen para

	llevar a cabo cada uno de los casos de uso descritos en el documento *Casos-Uso-Objeto*, supervisado por el analista de sistemas.
Objetivos	Relacionar una serie de controles con cada caso de uso.
Entradas	▪ Casos-Uso-Objeto ▪ Ejecutable de la aplicación
Producto generado	▪ Controles-Casos-Uso: Documento que contendrá los controles relacionados con las funcionalidades objeto de QA. Por cada control se deberá especificar lo siguiente: o Identificador: identificador del caso de uso, con el formato <*IdCasoUso*>-<*numSecuencial*>. o Tipo: tipo de control. o Descripción: breve descripción del control. o Imagen del control.

Tabla 20: Tarea ECGD2.2

Plantilla correspondiente al producto de salida Controles-Casos-Uso de la tarea ECGD2.2:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Controles-Casos-Uso

CONTROLES-CASOS-USO	
CASO DE USO	
IDENTIFICADOR	
TIPO CONTROL	
DESCRIPCIÓN CONTROL	
IMAGEN DEL CONTROL	
…	

Plantilla 10: Controles-Casos-Uso

Tarea ECGD2.3: Estudio de propiedades y eventos asociados a los controles objeto de QA

Tarea ECGD2.3	
Roles implicados	Analista de sistemas
Descripción	Estudiar las propiedades más relevantes de los controles extraídos en la anterior tarea. Verificar si existen eventos relacionados a determinadas acciones sobre los mencionados controles. En caso de existir eventos, verificar qué acciones llevan a cabo.
Objetivos	Conocer todas las propiedades relevantes de cada control así como los posibles eventos que se puedan disparar.
Entradas	▪ Controles-Casos-Uso ▪ Ejecutable de la aplicación ▪ Código aplicación
Producto generado	▪ Detalle-Controles: Documento en el que se debe plasmar por cada control: o Identificador: deberá ser el mismo identificador que el del correspondiente control en el documento *Controles-Casos-Uso*. o Tipo de control o Propiedades: · Nombre propiedad/evento · Valor propiedad · Descripción propiedad/evento

Tabla 21: Tarea ECGD2.3

Plantilla correspondiente al producto de salida Detalle-Controles de la tarea ECGD2.3:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Detalle-Controles

Detalle-Controles		
Detalle de controles de la aplicación objeto de QA		
Identificador:		
Tipo control:		
Propiedades/eventos:	Nombre	Valor
	Descripción	
	Nombre	Valor
	Descripción	
...		

Plantilla 11: Detalle-Controles

PAC - Plan de Aseguramiento de Calidad

Anotación didáctica

Una vez que se ha recopilado toda la información necesaria acerca de la aplicación objeto de aseguramiento de calidad, se deberán crear las pruebas pertinentes.

Para llevar a cabo pruebas de calidad sobre cualquier tipo de software, estas deberán ser diseñadas con pensamiento de encontrar errores en el mismo. Por ello, en esta fase se determinará de forma detallada el proceso que cualquier equipo de calidad debe seguir para la creación y ejecución de pruebas bien elaboradas.

El objetivo fundamental de esta fase es el de la creación de planes y casos prueba, verificar el correcto funcionamiento del software en cuestión y reporte de los posibles defectos existentes en el mismo al equipo de desarrollo.

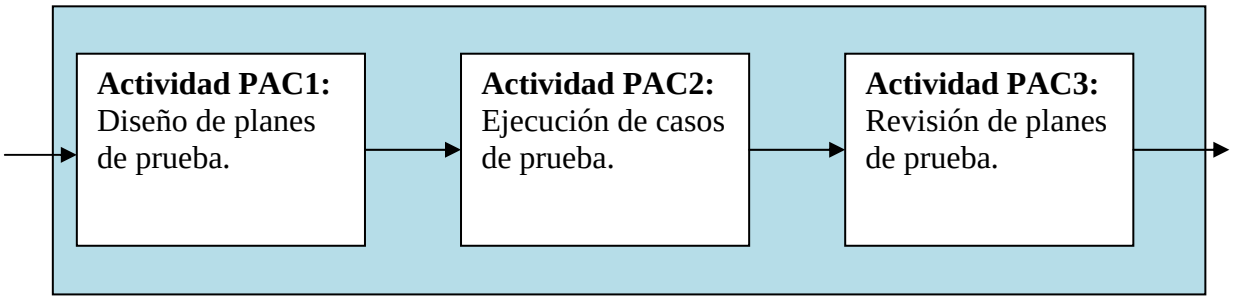


Ilustración 37: Diagrama temporal de actividades PAC

Actividad PAC1- Diseño planes de prueba

Actividad PAC1	
Descripción	Describir los objetivos, alcance, y enfoque del esfuerzo de testeo de software. Un plan de pruebas deberá identificar de forma global los casos de uso más frecuentes de un módulo funcional en concreto (descripción del plan de pruebas) así cómo el detalle de todas las pruebas a realizar sobre el mismo (casos de prueba), los cuales serán objeto de testeo para la certificación final del producto.
Objetivos	Determinación de un plan de pruebas que especifique minuciosamente el proceso con el que el equipo de calidad podrá verificar la calidad de la aplicación.

Tabla 22: Actividad PAC1

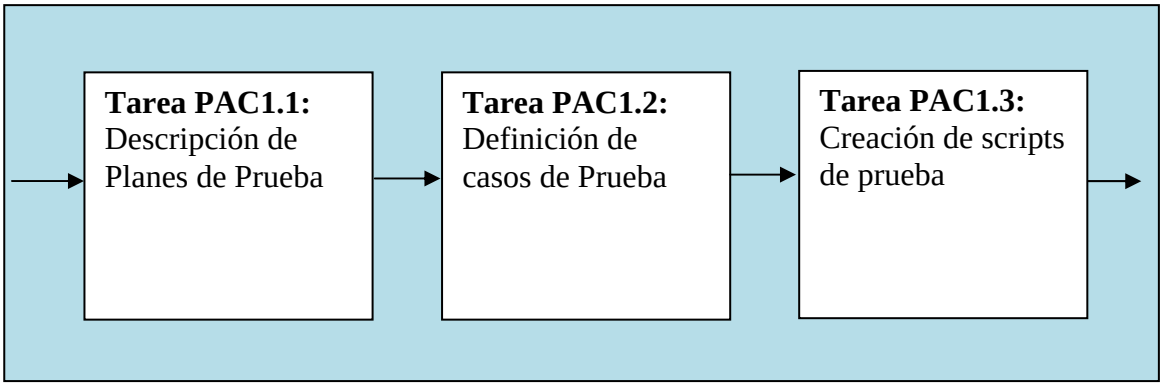


Ilustración 38: Diagrama temporal de tareas PAC1

Tarea PAC1.1: Descripción de Planes de Prueba

Tarea PAC1.1	
Roles implicados	JEC Analista de pruebas
Descripción	Crear planes de prueba de tal forma que cada módulo funcional que aparezca en los casos de uso objeto de QA, esté relacionado con alguno de mencionados planes. Opcionalmente, se puede subdividir un plan de pruebas por el enfoque que se pretenda dar a sus casos de prueba. El enfoque atenderá a criterios varios con motivo de permitir la división de trabajo en las futuras ejecuciones de los planes: - Subplanes con casos de prueba **positivos**. - Subplanes con casos de prueba que actúan intencionadamente de forma equivocada (**negativos**). - Subplanes con tipos de **controles** similares. - Subplanes con casos de prueba que verifican el comportamiento de la aplicación en casos extremos (**carga**, **concurrencia**…). Estudiar la documentación generada durante la fase ECGD, la cual proporcionará la información necesaria para definir **qué** funcionalidad/es se va/n a probar y **cómo** se va a hacer para verificar el correcto funcionamiento de la aplicación.
Objetivos	Obtención de los diferentes planes de prueba, cada uno de los cuales definirá la política que seguirán los casos de prueba contenidos en el mencionado plan.

Entradas	▪ Módulos-Funcionales ▪ Casos-Uso-Objeto ▪ Controles-Casos-Uso ▪ Detalle-Controles
Producto generado	▪ Planes-Prueba: En cada plan de pruebas se deberá especificar lo siguiente: o Información básica · Identificador: *<móduloFuncional>-<enfoque>* · Producto: *<nombreProducto><país><versión>* · Fecha de creación: *<dd-mm-yyyy>* · Fecha de revisión: *<dd-mm-yyyy>* · Modificado por. · Tiempo de ejecución. o Descripción: Información general acerca del alcance del plan de pruebas respecto a la funcionalidad del módulo. Se debe incluir un listado de los puntos principales incluidos en el testeo, correspondientes a los casos de prueba que se asociarán al plan. o Caso(s) de uso relacionado(s). o Enfoque de casos de prueba asociados al plan: Información acerca del enfoque general que se debe de aplicar en la definición de los casos de prueba contenidos en el plan (casos de prueba positivos, negativos, verifican situaciones extremas…). o Pruebas automáticas: Información acerca de las pruebas automáticas asociadas al plan.

Tabla 23: Tarea PAC1.1

Plantilla correspondiente al producto de salida Planes-Prueba de la tarea PAC1.1:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Planes-Prueba

Planes-Prueba	
Información básica	
Identificador	
Producto	
Fecha creación	
Fecha revisión	
Modificado por	
Tiempo ejecución	
Descripción	
Casos de uso relacionados	
Enfoque	
Pruebas automáticas	

Plantilla 12: Planes-Prueba

Tarea PAC1.2: Definición de casos de Prueba

Tarea PAC1.2	
Roles implicados	JEC Analista de pruebas
Descripción	Crear diferentes casos de prueba por cada plan de pruebas generado (al menos un caso de prueba por cada plan de prueba). Los casos de prueba que se crearán deben haber sido previamente definidos en el plan de pruebas, teniendo además que cumplir las especificaciones que en el mismo se recogen.
Objetivos	Obtención de los casos de prueba que serán llevados a cabo por parte de los Testers.
Entradas	▪ Controles-Casos-Uso ▪ Detalle-Controles ▪ Casos de uso
Producto generado	▪ Casos-Prueba: Todo caso de pruebas debe contener la siguiente información: o Identificador del caso de prueba: *<IdPlanPruebas>-<NombreFuncionalidad>-<numSecuencial>*. o Nombre. o Persona encargada: será la persona encargada de la ejecución de dicho caso de prueba. o Datos de partida: si fueran necesarios antes de

	empezar la ejecución propiamente dicha; o estado inicial en el que deberá encontrarse la aplicación. o Numerar claramente los pasos que se deben dar indicando qué campos rellenar y si fuera necesario, incluso los datos a introducir. o Tipo de ejecución: el tipo de ejecución podrá ser manual y/o automática. o Resultado esperado: imprescindible indicar qué se espera para considerar un resultado satisfactorio del caso.

Tabla 24: Tarea PAC1.2

Plantilla correspondiente al producto de salida Casos-Prueba de la tarea PAC1.2:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Casos-Prueba

Casos-Prueba	
Identificador	
Nombre	
Persona encargada	
Datos de partida	
Pasos	
Tipo ejecución	Manual ☐ Automática ☐
Resultado esperado	

Plantilla 13: Casos-Prueba

Tarea PAC1.3: Creación de scripts de Prueba

Tarea PAC1.3	
Roles implicados	Analista de pruebas Tester
Descripción	Habrá que verificar que todos y cada uno de los controles asociados a un caso de prueba que se pretenda automatizar, disponen de la propiedad *AutomationId*, lo cual puede verificarse en el documento *Detalle-Controles*. Si la mencionada propiedad no tiene valor, no se podrá automatizar el caso de prueba puesto que no habría forma de encontrar (mediante código) mencionado control. Crear los scripts correspondientes para llevar a cabo los casos de prueba que se hayan catalogado como automatizables en el documento *Casos-Prueba*. Almacenar los scripts generados de tal forma que puedan estar accesibles por cualquier persona del grupo de calidad que los pueda necesitar.
Objetivos	Obtención de los scripts necesarios para automatizar determinadas pruebas.
Entradas	▪ Casos-Prueba ▪ Detalle-Controles

Producto generado	▪ Script-Caso-Prueba: Este script podrá ser implementado en cualquier lenguaje que permita la comunicación con la aplicación en cuestión.

Tabla 25: Tarea PAC1.3

Actividad PAC2- Ejecución de casos de prueba

Actividad PAC2	
Descripción	Ejecución de los casos de prueba definidos de tal forma que se simule el comportamiento que tendrá el futuro usuario de la aplicación. Por esto, se deberán llevar a cabo en un entorno lo más similar posible al que tendrá el mencionado usuario.
Objetivos	Obtención de un resultado como consecuencia de la puesta en marcha de determinados casos de prueba, con el que posteriormente se podrá verificar si la aplicación tiene un correcto funcionamiento.

Tabla 26: Actividad PAC2

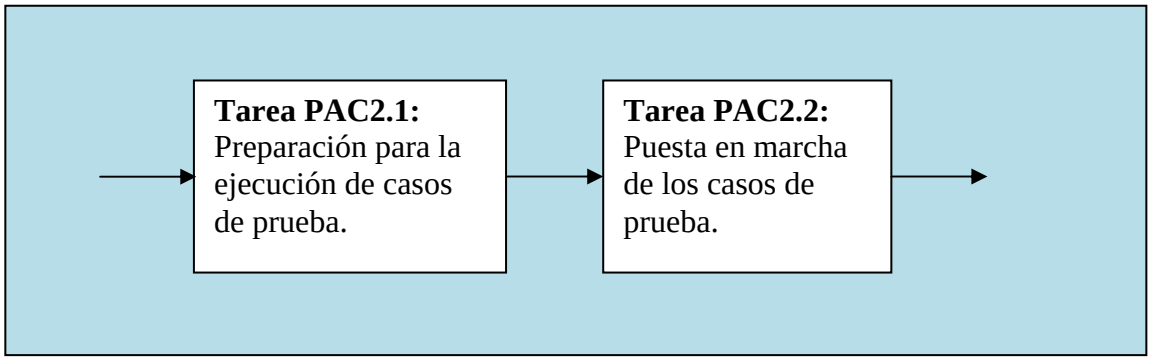


Ilustración 39: Diagrama temporal de tareas PAC2

Tarea PAC2.1: Preparación para la ejecución de casos de prueba

Tarea PAC2.1	
Roles implicados	Analista de Sistemas Tester
Descripción	Instalar en la medida de lo posible un entorno que simule el que tendrá un futuro usuario de la aplicación. Instalar la propia aplicación objeto de QA. En caso de tener que llevar a cabo pruebas automáticas, se deberá disponer de los correspondientes scripts.
Objetivos	Disponer de todo lo necesario para llevar a cabo los casos de prueba.
Entradas	▪ Software necesario ▪ Instalable aplicación ▪ Script-Caso-Prueba ▪ Planes-Prueba ▪ Casos-Prueba
Producto generado	▪ Entorno-Pruebas: Documento que contendrá las especificaciones del sistema con las que se llevarán a cabo las pruebas en cuestión. Dicho documento deberá contener lo siguiente: o Especificaciones hardware: procesador, velocidad,

	memoria RAM. o Especificaciones software: sistema operativo y software adicional necesario, incluyendo la versión.

Tabla 27: Tarea PAC2.1

Plantilla correspondiente al producto de salida Entorno-Pruebas de la tarea PAC2.1:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Entorno-Pruebas

Entorno-Pruebas			
Especificaciones HW			
Procesador			
Velocidad			
Memoria RAM			
Especificaciones SW			
Sistema operativo			
Software necesario		**Versión**	
Software necesario		**Versión**	
…			

Plantilla 14: Entorno-Pruebas

Tarea PAC2.2: Puesta en marcha de los casos de prueba

Tarea PAC2.2	
Roles implicados	Tester
Descripción	Ejecutar la aplicación en cuestión. Antes de llevar a cabo un caso de prueba se deberá dejar la aplicación en un estado inicial válido para llevar a cabo la puesta en marcha de la prueba. En caso de no ser un caso de prueba automatizable, se deberán poner en marcha cada uno de los pasos especificados en el correspondiente documento *Casos-Prueba*. En caso de que se trate de un caso de prueba automatizable, se deberá ejecutar el script en cuestión. Capturar el resultado obtenido como consecuencia de la ejecución del caso de prueba para que posteriormente se verifique si ha habido éxito en la ejecución del mismo.
Objetivos	Obtener los resultados de los casos de prueba.
Entradas	▪ Ejecutable de la aplicación. ▪ Casos-Prueba ▪ Script-Caso-Prueba
Producto generado	▪ Resultado-Casos-Prueba: Este producto deberá contener los siguientes campos: o Identificador del caso de prueba.

	o Resultado esperado. o Resultado obtenido. o Conclusiones: este campo será cumplimentado en posteriores actividades. o Resolución: este campo será cumplimentado en posteriores actividades.

Tabla 28: Tarea PAC2.2

Plantilla correspondiente al producto de salida Resultado-Casos-Prueba de la tarea PAC2.2:

Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Resultado-Casos-Prueba

Resultado-Casos-Prueba	
Id. Caso de prueba	
Resultado esperado	
Resultado obtenido	
Conclusiones	
Resolución	**Correcto** ☐ **Incorrecto** ☐

Plantilla 15: Resultado-Casos-Prueba

Actividad PAC3- Revisión de planes de prueba

Actividad PAC3	
Descripción	Revisión de los casos de prueba ejecutados para poder certificar la calidad de la aplicación. Cualquier defecto encontrado deberá ser reportado al equipo de desarrollo para que tome las medidas oportunas.
Objetivos	Verificar si se ha encontrado algún tipo de defecto en la aplicación.

Tabla 29: Actividad PAC3

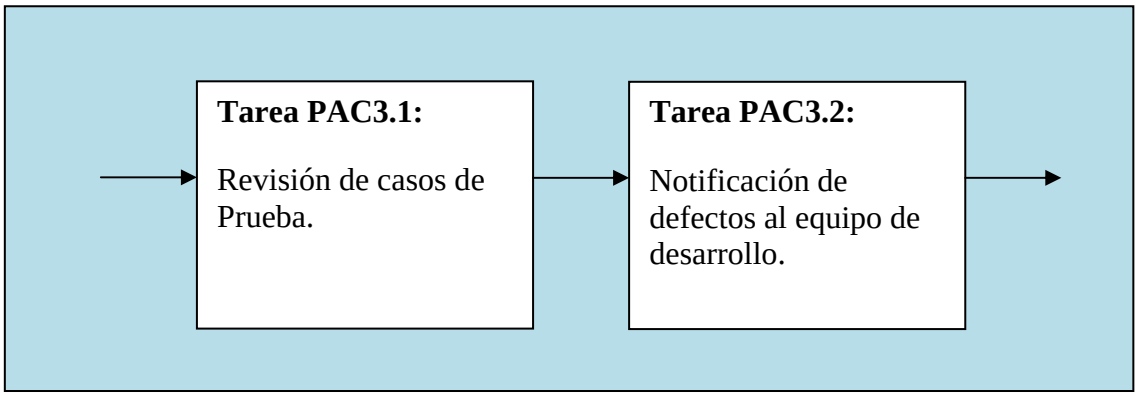


Ilustración 40: Diagrama temporal de tareas PAC3

Tarea PAC3.1: Revisión de casos de Prueba

Tarea PAC3.1	
Roles implicados	Tester
Descripción	Verificar si el resultado obtenido (Resultado-Casos-Prueba) como consecuencia de la ejecución de un determinado caso de prueba es el que se esperaba.
Objetivos	Constatar de forma formal qué partes de la funcionalidad de una aplicación funcionan correctamente y cuáles deben ser analizadas debido a un mal funcionamiento.
Entradas	▪ Resultado-Casos-Prueba
Producto generado	▪ Resultado-Casos-Prueba: Rellenar los siguientes campos del correspondiente documento *Resultado-Casos-Prueba*: o Conclusiones: conclusión que se obtiene sobre la ejecución del caso de prueba. o Resolución: pueden tomarse dos valores *correcto* o *incorrecto*.

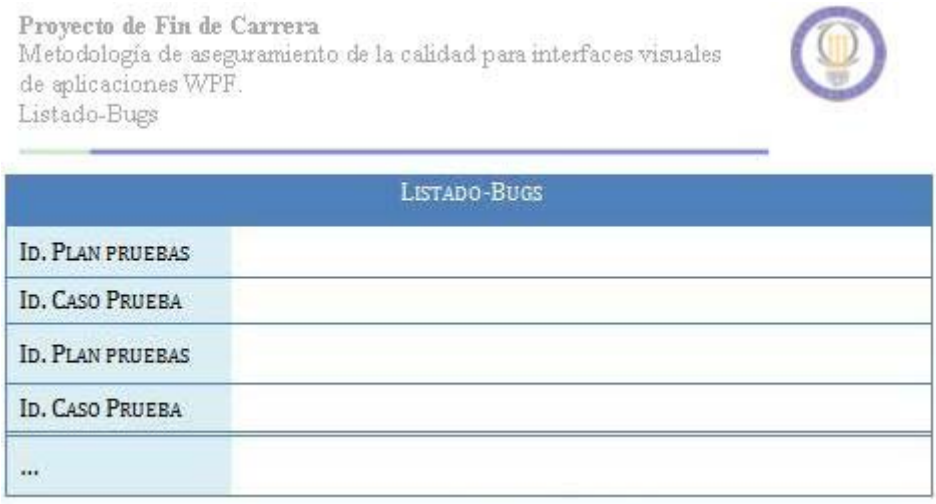
Tabla 30: Tarea PAC3.1

Tarea PAC3.2: Notificación de defectos al equipo de desarrollo

Tarea PAC3.1	
Roles implicados	JEC Representante del equipo de desarrollo Tester
Descripción	Crear un listado por parte del tester con los casos de prueba que han experimentado un comportamiento no deseado. El JEC deberá revisar y posteriormente reportar dicho listado al representante del equipo de desarrollo para que tome las medidas oportunas.
Objetivos	Dar a conocer al responsable del equipo de desarrollo los defectos encontrados en la aplicación.
Entradas	▪ Resultado-Casos-Prueba
Producto generado	▪ Listado-Bugs: Este documento contendrá una lista con los diferentes defectos encontrados en la fase de pruebas.

Tabla 31: Tarea PAC3.1

Plantilla correspondiente al producto de salida Listado-Bugs de la tarea PAC3.2:


Proyecto de Fin de Carrera
Metodología de aseguramiento de la calidad para interfaces visuales de aplicaciones WPF.
Listado-Bugs

LISTADO-BUGS	
ID. PLAN PRUEBAS	
ID. CASO PRUEBA	
ID. PLAN PRUEBAS	
ID. CASO PRUEBA	
...	

Plantilla 16: Listado-Bugs

3. Planificación y Presupuesto de la metodología

A continuación se expone el presupuesto y planificación necesarios para llevar a cabo la creación de la metodología:

3.1. Desglose por fases

A continuación se presenta las horas totales de cada una de las fases seguidas en la creación de la metodología.

Fase\Horas	Horas totales
Estudio WPF	40
Estudio metodologías existentes	30
Creación metodología	80
Total horas	150

Tabla 32: horas/fase metodología

3.2. Planificación

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Creación de la metodología de calidad**	**30 días**	**vie 14/11/08**	**mié 10/12/08**
2	**Estudio WPF**	**8 días**	**vie 14/11/08**	**jue 20/11/08**
3	Estudio plataforma .NET	3,2 días	vie 14/11/08	lun 17/11/08
4	Estudio controles y XAML	4,8 días	mar 18/11/08	jue 20/11/08
5	Estudio metodologías existentes	6 días	vie 21/11/08	mié 26/11/08
6	**Creación metodología**	**16 días**	**mié 26/11/08**	**mié 10/12/08**
7	Esquema y planteamiento de la metodología	1,6 días	mié 26/11/08	jue 27/11/08
8	Redacción de la metodología	14,4 días	jue 27/11/08	mié 10/12/08

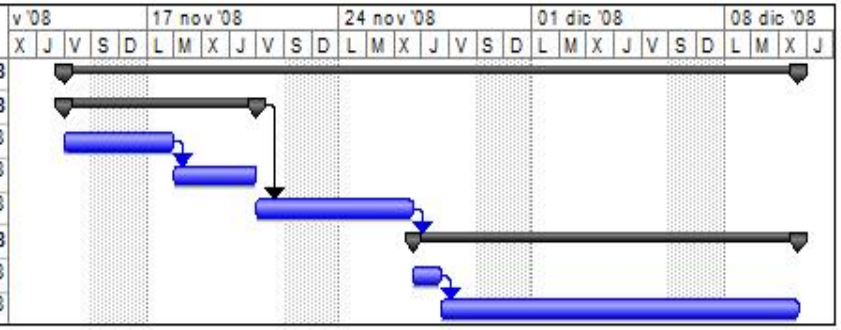


Ilustración 41: Gantt metodología

3.3. Salarios por categoría

Para conocer el gasto de la empresa en relación al trabajo de sus empleados o personal que conforma la misma es necesario conocer los salarios de los trabajadores:

Cargo	Sueldo Bruto Año	Sueldo Bruto mes	Coste Hora/empleado
Analista	30.000 €	2.500 €	15,625 €
Especialista Metodologías	19.000 €	1.583,33 €	9,895 €

Tabla 33: Salarios por categoría metodología

A continuación, tras conocer los salarios de los empleados hay que verificar cuantas horas del proyecto va a dedicar cada uno de ellos, en relación a las distintas fases del mismo, incluyendo los costes asociados:

Fase	Analista (horas)	Especialista metodologías (horas)	Total (€)
Estudio WPF	30	10	567,7
Estudio metodologías existentes	10	20	354,15
Creación metodología	40	40	1.020,8
TOTAL	**80*15,625=1.250 €**	**70*9,895=692,65 €**	**1.942,65**

Tabla 34: Coste salarios de personal/fase metodología

3.4. Gastos de personal imputables al proyecto

A partir de la tabla anterior es posible sacar la relación de costes de los empleados del proyecto en relación a las horas totales que va a dedicar cada uno de ellos al mismo:

Empleado	Horas	Coste
Analista	80 h	80*15,625=1.250 €
Especialista Metodologías	70 h	70*9,895=692,65 €
TOTAL	**150 h**	**1.942,65 €**

Tabla 35: Coste horas/empleados metodología

3.5. Resumen del presupuesto

En esta tabla se especifican todos los gastos obtenidos anteriormente, para la presentación del resumen del presupuesto, al cual debe incluirse el IVA:

Gasto	Coste
Empleados	1.942,65 €
IVA (16%)	314,024 €
TOTAL	**2.256,674 €**

Tabla 36: Resumen presupuesto metodología

El presupuesto final de la creación de la metodología de pruebas sobre interfaces visuales de aplicaciones WPF asciende a un TOTAL de **2.256,674 €** (DOS MIL DOSCIENTOS CINCUENTA Y SEIS, CON SEISCIENTOS SETENTA Y CUATRO EUROS) **IVA INC.**

IV

Aplicación WPF

IV. Aplicación WPF

1. Documento de análisis del sistema

A continuación se expone el documento de análisis de sistema, en el cual se recopilará la información necesaria que debe satisfacer la aplicación que se pretende construir.

1.1. Introducción

Se desea realizar un administrador de contenidos (Content Management System en inglés, abreviado CMS). En este caso su temática se centrará en un marco cinematográfico.

Se trata de una aplicación que permita crear una estructura de soporte para la creación y administración de contenidos por parte de los participantes. Dicha aplicación será visual, por lo que dispondrá de una interfaz que a su vez controlará una base de datos, donde se aloja el contenido del sitio.

Con la elaboración de este documento se pretende determinar formalmente las necesidades que debe satisfacer la aplicación, quedando reflejados los requisitos de usuario de la misma.

1.2. Propósito del documento

El propósito que se desea alcanzar con la elaboración de este documento es ofrecer una visión estricta y formal del alcance del software. El documento contiene todos los requisitos de usuario capturados, permitiendo consultar de forma rápida y eficaz cualquier problema que pueda surgir, tales como contradicciones entre requisitos, problemas legales, etc.

El documento está dirigido al cliente, y tiene como objetivo dejar constancia de que el mismo está de acuerdo con las funcionalidades de la aplicación, para ello el propio cliente debe validar los requisitos de usuario capturados en este documento.

También está orientado para que sirva como referencia para el personal que trabaja en el desarrollo del producto, como pueden ser el jefe de proyecto, los analistas o los programadores, sirviendo de documento de consulta para la elaboración del producto software.

1.3. Alcance del software

La aplicación *Gestiona* proporciona la funcionalidad de poder añadir, eliminar o modificar los diferentes tipos de contenido, permitiendo búsquedas sobre el mismo con el objetivo de facilitar la recuperación de la información que se requiera en cada momento. Para llevar a cabo cualquier tipo de acción sobre la aplicación se deberá estar dado de alta previamente en la misma, por lo que en todos los casos será necesario introducir un nombre de usuario y contraseña.

Los beneficios obtenidos del uso de la aplicación son disponer y administrar **diferentes tipos de contenido**, el cual puede haber sido introducido por el usuario. No sólo será de gran utilidad el disponer de determinado contenido, sino de que además éste sea **fácilmente recuperable** por parte del usuario y se puedan llevar a **acciones sobre el mismo**.

1.4. Visión general del documento

Los diferentes apartados en los que centrará el documento son los siguientes:

- Capacidades generales del producto.
- Requisitos de usuario.

1.5. Capacidades generales del producto

El cliente necesita un software que le permita manejar contenido de ámbito cinematográfico de forma fácil, intuitiva y sencilla. Por ello el software tiene las siguientes capacidades principales:

- Permite **añadir películas** a la base de datos, pudiendo especificar cada uno de los diferentes campos de los que se compone una película.

- Permite **modificar** las **películas** existentes en la base de datos, pudiendo hacerlo sobre los campos de las mismas que corresponda.

- Permite **eliminar** las **películas** existentes en la base de datos.

- Permite **recuperar películas** de la base de datos en función a determinados parámetros de búsqueda.

- Permite **puntuar** las **películas** con una valoración entre 1 y 5 estrellas, por lo que con cada votación se irá actualizando la valoración media de la película.

- Permite diferentes **opciones de visualización** del contenido.

1.6. Requisitos de usuario

En este apartado se describen los requisitos de usuario, que son la base de todo desarrollo de software. Para ello, se han utilizado algunas de las especificaciones que la ESA propone para definición de requisitos de usuario.

Cada requisito debe ser identificado unívocamente por su ID, en este caso la numeración será URC-X (Requisito de usuario de capacidad número X) y URR-X (Requisito de usuario de restricción número X).

Algunos requisitos tienen más prioridad que otros, dependiendo por ejemplo de si ciertas partes del software se empiezan a desarrollar antes que otras; en ese caso, esos requisitos deben tener más prioridad. Por tanto, cada requisito está marcado con una medida de prioridad.

También debe especificarse la estabilidad de los requisitos, pudiendo ser estables o inestables. Un requisito inestable es aquel que tiene cierta posibilidad de ser modificado en el futuro, por problemas económicos, tecnológicos o legales, por ejemplo.

La fuente de cada requisito deberá ser indicada. Según la ESA, se puede definir usando el identificador de un requisito del sistema, una referencia recíproca del documento o incluso el nombre de una persona o grupo. En general para este desarrollo, los requisitos han sido obtenidos básicamente de las especificaciones dadas por el propio cliente, o del equipo de desarrollo en el caso en el que lo considere conveniente.

Finalmente habrá otro campo denominado *necesidad*, el cual indicará el grado de importancia que tiene la implementación de un determinado requisito. Los valores que puede tomar dicho campo serán: esencial, conveniente y opcional.

Lo especificado anteriormente se trata de los campos que deberá contener la tabla donde se exponga cada requisito. A continuación se exponen una serie de **recomendaciones** a partir de las cuales se deben crear los diferentes requisitos de usuario. Estas pueden ser muy útiles, puesto que en muchas ocasiones cuando se llega a una fase más avanzada del proyecto, surge el problema de que algunos de ellos son inconsistentes, no alcanzables razonablemente, etc.:

- Consistencia: comprobar que los requisitos no sean lógicamente incompatibles o contradictorios entre sí. Un conjunto de requisitos contradictorio no es implementable.

- Factibilidad: comprobar si los requisitos pueden ser llevados a cabo sobre el sistema en cuestión.

- Requisitos razonables: debe haber un equilibrio en cuanto a determinados parámetros tales como rendimiento, fiabilidad o uso de memoria entre otros.

- Verificables: un requisito es verificable si existe un proceso finito y no costoso para demostrar que el sistema cumple con el requisito. Un requisito ambiguo no es, en general, verificable. Requisitos como "la interfaz es amigable" o "normalmente debe hacer…" no son válidos porque términos como "amigable" o "normalmente" no son objetivos, sino subjetivos y ambiguos. Sin embargo, un requisito como: "el programa debe dar la salida para el proceso X por debajo de 10 segundos el 60% de las veces" es válido porque es verificable.

En función de todo lo descrito anteriormente, se ha creado una tabla genérica para los requisitos (tanto para los de restricción como para los de capacidad) en la que se incluyen todos los aspectos definidos:

ID:	**Prioridad:**	**Estabilidad:**	**Necesidad:**
Descripción:			
Fuente:			

Tabla 37: Formato requisitos de usuario

1.6.1. Requisitos de usuario de capacidad

ID: URC-01	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá acceder al contenido ofrecido por la aplicación una vez que se autentique en la misma.			
Fuente: cliente			

ID: URC-02	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá acceder a la aplicación introduciendo su login y password.			
Fuente: cliente			

ID: URC-03	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá salir de la aplicación.			
Fuente: cliente			

ID: URC-04	**Prioridad:** 2ª	**Estabilidad:** Estable	**Necesidad:** Conveniente
Descripción: El usuario podrá verificar una lista de los tres usuarios que más contenido aportan al sistema.			
Fuente: cliente			

ID: URC-05	**Prioridad:** 2ª	**Estabilidad:** Estable	**Necesidad:** Conveniente
Descripción: El usuario podrá modificar la vista de la lista de los tres usuarios que más contenido aportan al sistema, de tal forma que aparezcan más o menos datos de cada uno de ellos.			
Fuente: cliente			

ID: URC-06	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: Una película debe disponer de los siguientes campos: ▪ Nombre. ▪ Cartel: imagen del cartel de la película. ▪ Año de estreno. ▪ Nacionalidad. ▪ Valoración. ▪ Tipo de película. ▪ Argumento. ▪ Dirección: director de la película. ▪ Guión: guionista de la película. ▪ Actores principales: protagonista masculino y protagonista femenino.			
Fuente: cliente			

ID: URC-07	**Prioridad:** 2ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá verificar la lista de las diez películas más vistas por todos los usuarios.			
Fuente: cliente			

ID: URC-08	**Prioridad:** 2ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá modificar la vista de la lista de las diez películas más vistas, de tal forma que aparezcan más o menos datos de cada una de ellas.			
Fuente: cliente			

ID: URC-09	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá añadir películas a la base de datos.			
Fuente: cliente			

ID: URC-10	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: Al insertar una película en la base de datos, todos los campos son opcionales salvo el de *nombre de la película*, que además no deberá existir en la misma.			
Fuente: equipo desarrollador			

ID: URC-11	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá modificar todos y cada uno de los campos referentes a una determinada película.			
Fuente: cliente			

ID: URC-12	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: Al modificar el nombre de una película de la base de datos, este debe tener al menos un carácter, y no debe existir en la misma.			
Fuente: equipo desarrollador			

ID: URC-13	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá eliminar las películas existentes en la base de datos.			
Fuente: cliente			

ID: URC-14	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá recuperar películas de la base de datos en función a determinados parámetros de búsqueda.			
Fuente: cliente			

ID: URC-15	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El sistema deberá proporcionar una vista resumen de las películas encontradas como consecuencia de una búsqueda por parte del usuario.			
Fuente: cliente			

ID: URC-16	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá personalizar los campos a mostrar de la vista resumen de las películas.			
Fuente: cliente			

ID: URC-17	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: En la vista resumen de las películas siempre se mostrará el cartel de la misma, independientemente del resto de campos a mostrar.			
Fuente: equipo desarrollador			

ID: URC-18	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: En usuario podrá visualizar todos los campos referentes a una película en una vista ampliada.			
Fuente: cliente			

ID: URC-19	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: En usuario podrá cerrar una película que previamente ha sido visualizada.			
Fuente: cliente			

ID: URC-20	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El usuario podrá puntuar las películas.			
Fuente: cliente			

ID: URC-21	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El campo *Valoración* de una película debe contener valores enteros en el rango del 1 al 5.			
Fuente: cliente			

ID: URC-22	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: Cuando un usuario puntúa una determinada película, se actualizará automáticamente con la puntuación media de la misma.			
Fuente: cliente			

ID: URC-23	**Prioridad:** 2ª	**Estabilidad:** Estable	**Necesidad:** Conveniente
Descripción: El sistema deberá proporcionar una respuesta de aceptación (feedback) para cada función realizada.			
Fuente: equipo desarrollador			

ID: URC-24	**Prioridad:** 2ª	**Estabilidad:** Estable	**Necesidad:** Conveniente
Descripción: El sistema deberá proporcionar al usuario mensajes de confirmación ante acciones críticas sobre datos del sistema.			
Fuente: equipo desarrollador			

ID: URC-25	**Prioridad:** 1ª	**Estabilidad:** No estable	**Necesidad:** Esencial
Descripción: El campo *Tipo de película* de una película debe ser de alguno de los siguientes: ▪ Acción. ▪ Comedia. ▪ C. Ficción. ▪ Drama. ▪ Suspense. ▪ Terror.			
Fuente: cliente			

ID: URC-26	**Prioridad:** 1ª	**Estabilidad:** No Estable	**Necesidad:** Esencial
Descripción: Los parámetros de búsqueda que se utilizarán como filtros a la hora de realizar consultas sobre películas en la base de datos deberán ser: ▪ Nombre. ▪ Tipo de película. ▪ Año.			
Fuente: cliente			

ID: URC-27	**Prioridad:** 1ª	**Estabilidad:** No Estable	**Necesidad:** Esencial
Descripción: Los campos que componen la vista resumen de cada película como resultado de una búsqueda pueden ser: ▪ Nombre. ▪ Año. ▪ Director. ▪ Actores principales (actor y actriz).			
Fuente: cliente			

1.6.2. Requisitos de usuario de restricción

ID: URR-01	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El sistema deberá funcionar en sistemas operativos Windows XP y Windows Vista.			
Fuente: cliente			

ID: URR-02	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El sistema no debe requerir conexión a Internet para funcionar correctamente.			
Fuente: cliente			

ID: URR-03	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: El tiempo de respuesta en cualquier tipo de acción llevada a cabo por el usuario no debe ser superior a 1 segundo.			
Fuente: cliente			

ID: URR-04	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: Los colores utilizados en la interfaz de usuario no deben perjudicar la visibilidad del contenido que se muestra.			
Fuente: equipo desarrollador			

ID: URR-05	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: La aplicación deberá estar programada en WPF (Windows Presentation Foundation).			
Fuente: cliente			

ID: URR-06	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: Los controles de la aplicación deberán ser identificables.			
Fuente: equipo desarrollador			

ID: URR-07	**Prioridad:** 1ª	**Estabilidad:** Estable	**Necesidad:** Esencial
Descripción: Acceso permitido a la interfaz de usuario mediante programación.			
Fuente: equipo desarrollador			

2. Documento de diseño del sistema

En este documento se exponen diferentes modelos necesarios para facilitar la creación de la aplicación en cuestión. En cada uno de ellos se muestran diferentes aspectos de diseño de la misma.

2.1. Propósito del documento

El propósito del documento es trazar a grandes rasgos el diseño de la aplicación. El documento va dirigido principalmente a los desarrolladores de la aplicación y a los encargados del mantenimiento del software. Los desarrolladores se basarán en este documento para implementar la arquitectura final del software, siguiendo las directrices dadas aquí.

2.2. Visión general del documento

Los diferentes apartados en los que centra el documento son los siguientes:

- **Modelo de clases:** se expondrán las clases involucradas, mostrando las relaciones existentes entre ellas.
- **Modelo Visual:** se expondrá la jerarquía visual de controles que componen la aplicación.
- **Modelo de casos de uso:** se expondrán en forma de casos de uso las funcionalidades más importantes de la aplicación.
- **Modelo de datos:**
  - Modelo entidad-relación.
  - Modelo relacional.

2.3. Modelo de clases

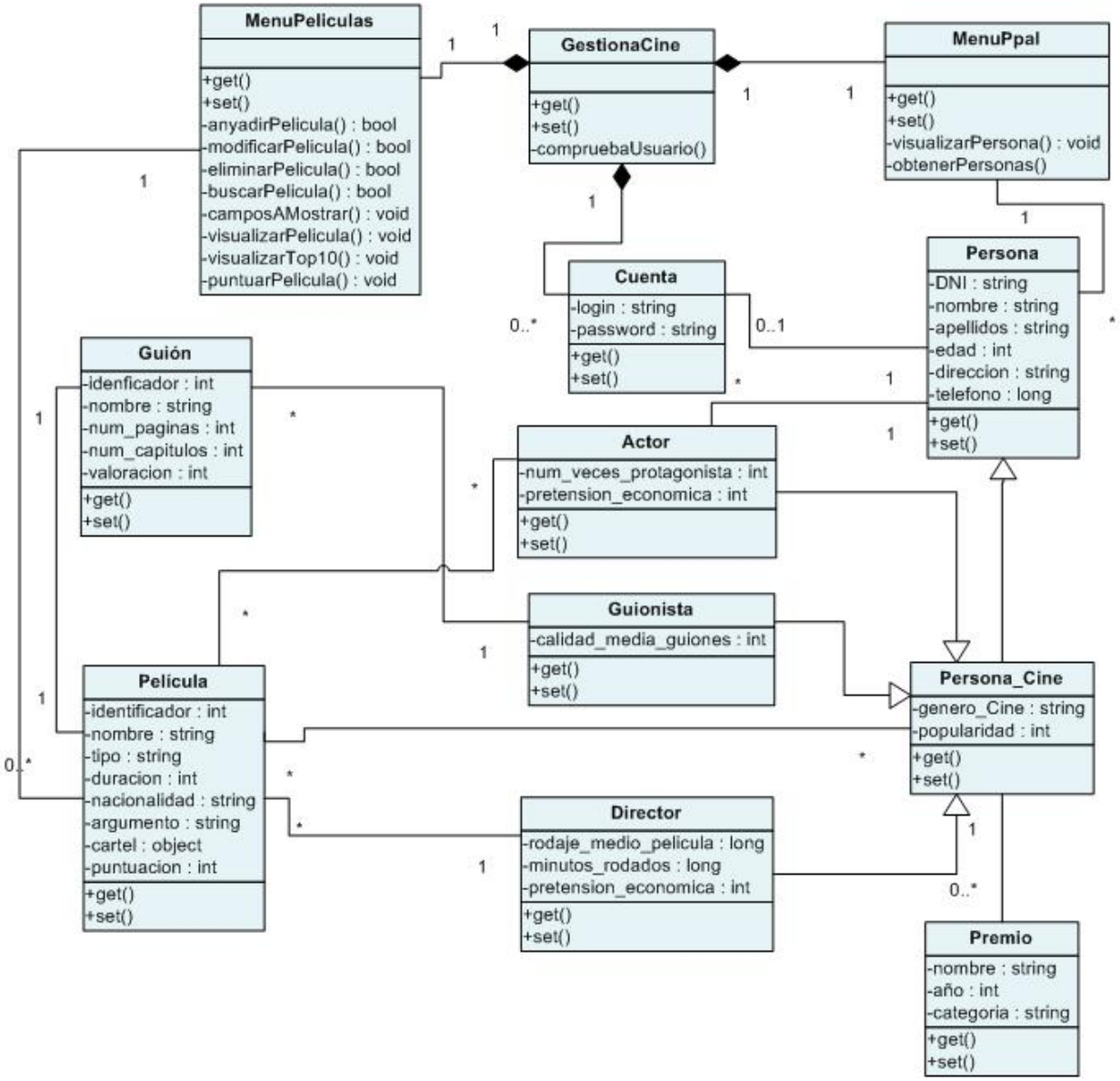


Ilustración 42: Modelo de clases

2.4. Modelo Visual

Al tratarse de una aplicación visual, en este documento de diseño se deberá especificar qué bloques principales deberá tener la misma en cuanto a visualización. Con ello no se pretende diseñar la interfaz de usuario de la aplicación sino más bien identificar los controles más importantes que deberán tener lugar, así cómo la distribución de los mismos, pudiendo en un futuro introducir otros nuevos que también puedan ser necesarios.

La elección de los diferentes controles que se expondrán más abajo, ha sido tomada en función a la funcionalidad requerida por parte de la aplicación, apoyándose por ello en el documento de requisitos. Por ello, a través del modelo visual se debería poder llevar a cabo cualquier funcionalidad de las especificadas en los mencionados requisitos.

A continuación se expondrán los diferentes árboles de controles catalogados por su pertenencia a cada interfaz de la aplicación. En este caso habrá tres interfaces principalmente, las cuales son:

- **Window1:** interfaz de usuario de login de usuario.
- **MenuPpal:** interfaz de usuario principal de la aplicación.
- **Películas:** interfaz de usuario de películas.

2.4.1. Window1

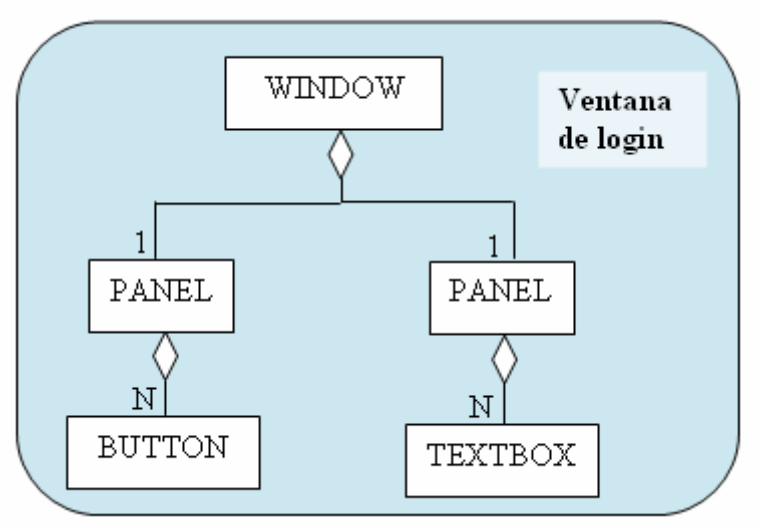


Ilustración 43: Window1

2.4.2. MenuPpal

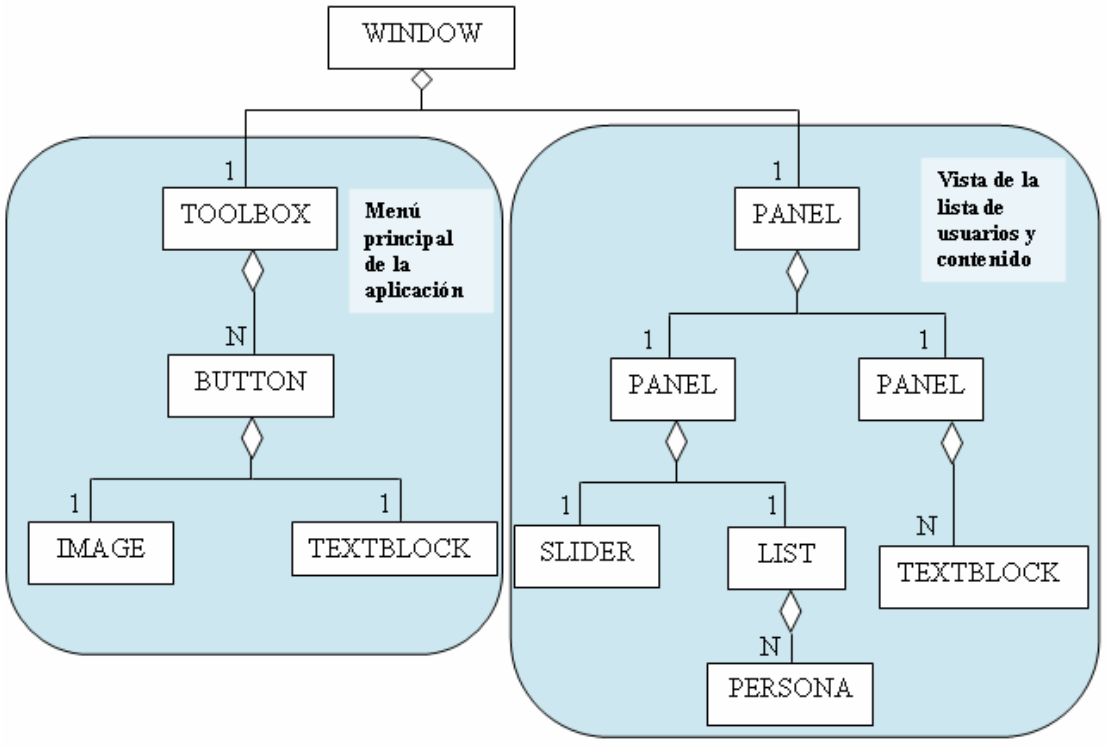


Ilustración 44: MenuPpal

2.4.3. Películas

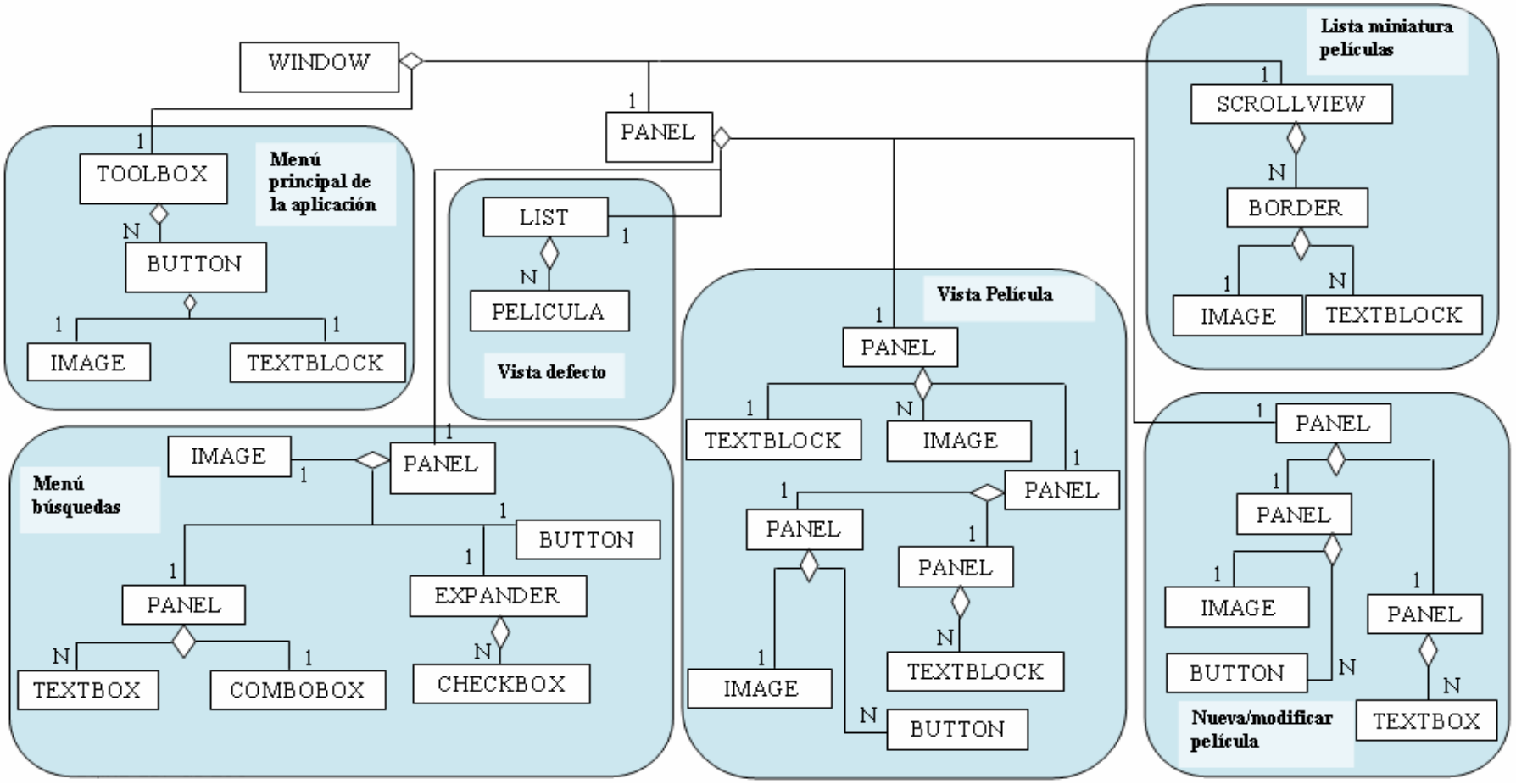


Ilustración 45: Películas

2.5. Modelo de casos de uso

Todos los casos de uso se describirán de una forma concisa y descriptiva. Para proporcionar una mayor claridad, estos serán catalogados por el tipo de funcionalidad que desempeñan, definidas posteriormente. Para mostrar los casos de uso se empleará una tabla a modo de plantilla que contiene la siguiente información:

- **Nombre:** nombre que identificará el caso de uso.
- **Actores:** los posibles roles de usuario que pueden intervenir en el caso de uso.
- **Objetivo:** la descripción de la finalidad que pretende el actor con la realización de este caso de uso.
- **Descripción:** breve explicación del proceso llevado a cabo en el caso de uso.
- **Precondiciones:** son aquellos hechos que deben ser previamente ciertos para poder llevar a cabo el caso de uso.
- **Poscondiciones:** son aquellos hechos que la ejecución del caso de uso hace verdaderos o ciertos.
- **Escenario:** descripción esquemática de las fases que componen el caso de uso.
- **Escenario Alternativo:** describe una secuencia alternativa a la expresada en el escenario básico.

Como se ha comentado anteriormente, se ha decidido catalogar los casos de uso por funcionalidades, las cuales serán:

- **Operaciones de acceso:** este grupo de la clasificación es el que permitirá al usuario acceder al sistema y salir del mismo cuando lo desee.

- **Gestión de operaciones e interacción con el sistema:** este grupo de la clasificación es el que permitirá al usuario llevar a cabo las operaciones ofrecidas por la aplicación.

2.5.1. Operaciones de Acceso

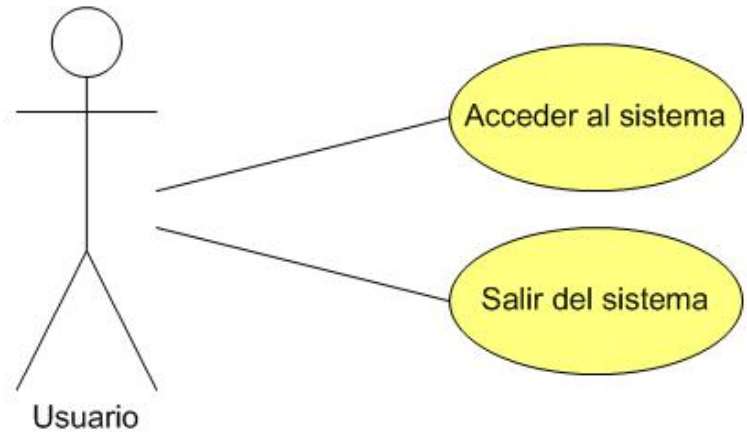


Ilustración 46: Operaciones de Acceso

CUOA-01	ACCEDER AL SISTEMA
Nombre:	Acceder al sistema
Actores:	Usuario.
Objetivo:	Acceder a la aplicación por parte del usuario.
Descripción:	El usuario se autentica en el sistema para acceder a la funcionalidad propia de usuarios registrados.
Precondiciones:	Estar registrado en el sistema.
Poscondiciones:	El usuario estará autenticado en el sistema
Escenario:	1. El usuario accede al formulario de identificación. 2. Introduce nombre de usuario y contraseña para acceder al sistema.
Escenario Alternativo:	**1, 2:** El usuario puede abandonar el proceso de acceso al sistema en cualquier momento. **2:** Si el usuario no introduce correctamente su nombre de usuario o contraseña se le denegará el acceso, volviendo al formulario anteriormente mencionado.

CUOA-02	SALIR DEL SISTEMA
Nombre:	Salir del sistema.
Actores:	Usuario.
Objetivo:	Salir de la aplicación por parte del usuario.
Descripción:	El usuario sale del sistema, pero podrá volver a entrar puesto que su cuenta permanece creada.
Precondiciones:	Estar autenticado en el sistema.
Poscondiciones:	El usuario ha salido del sistema.
Escenario:	1. El usuario pulsa en la opción salir en el menú principal.
Escenario Alternativo:	**1:** El usuario puede abandonar el proceso abandonar el sistema en cualquier momento.

2.5.2. Gestión de operaciones e interacción con el sistema

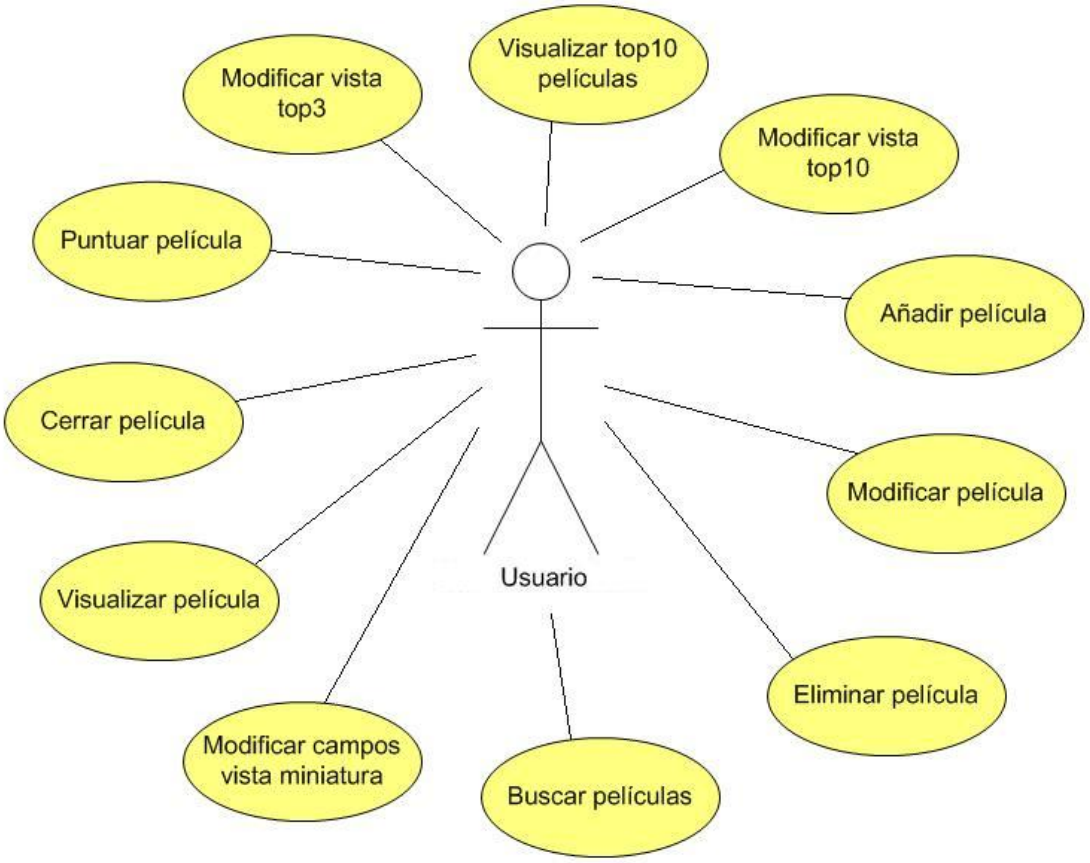


Ilustración 47: Gestión de operaciones e interacción con el sistema

CUOI-01	VISUALIZAR TOP-10 PELÍCULAS
Nombre:	Visualizar TOP-10 películas
Actores:	Usuario.
Objetivo:	Visualizar la lista TOP-10 de películas.
Descripción:	El usuario podrá visualizar la lista contenedora de las diez películas más vistas por parte de los usuarios.
Precondiciones:	Estar autenticado en el sistema.
Poscondiciones:	Se mostrará una lista con el TOP-10 de las películas.
Escenario:	1. El usuario accede al apartado de películas. 2. El usuario deberá cerrar cualquier contenido que haya abierto previamente del panel principal.
Escenario Alternativo:	**1, 2:** El usuario puede abandonar el proceso de visualización del TOP-10 en cualquier momento.

CUOI-02	MODIFICAR VISTA TOP-10
Nombre:	Modificar vista TOP-10
Actores:	Usuario.
Objetivo:	Modificar la vista de la lista TOP-10 de películas.
Descripción:	El usuario podrá cambiar los campos que se muestran en la lista contenedora de las diez películas más vistas por parte de los usuarios.
Precondiciones:	Estar autenticado en el sistema.
Poscondiciones:	Se actualizará la vista de la lista del TOP-10 de películas con los campos que proceda.
Escenario:	1. El usuario deberá hacer **visible la lista TOP-10** de películas. 2. El usuario deberá seleccionar otra vista de las que se proporcionen.
Escenario Alternativo:	**1, 2:** El usuario puede abandonar el proceso de modificación de la vista del TOP-10 en cualquier momento.

CUOI-03	AÑADIR PELÍCULA
Nombre:	Añadir película.
Actores:	Usuario.
Objetivo:	Añadir una película en la base de datos.
Descripción:	El usuario podrá añadir una película en la base de datos.
Precondiciones:	Estar autenticado en el sistema.
Poscondiciones:	La base de datos contendrá una nueva película.
Escenario:	1. El usuario accede al apartado de películas. 2. El usuario accede al formulario de creación de una nueva película a través del botón correspondiente. 3. El usuario cumplimenta los datos requeridos en el formulario, siendo obligatorio introducir el nombre de la película. Finalmente pulsará el botón de guardar.
Escenario Alternativo:	**1, 2, 3:** El usuario puede abandonar el proceso de creación de la película en cualquier momento. **3:** Si el usuario no introduce todos los datos necesarios, o bien el nombre de la película ya está en la base de datos, volverá a aparecer el formulario indicando qué campo no es correcto.

CUOI -04	MODIFICAR PELÍCULA
Nombre:	Modificar película
Actores:	Usuario.
Objetivo:	Modificar una película de la base de datos.
Descripción:	Un usuario puede modificar los campos de una película por los motivos que desee.
Precondiciones:	Estar autenticado en el sistema y que la película exista en la base de datos.
Poscondiciones:	Se modificará la película de la base de datos.
Escenario:	1. El usuario accede al apartado de películas. 2. El usuario **visualiza la película** que desea modificar. 3. El usuario pulsa la opción modificar de la propia película. 4. El usuario puede modificar cualquier campo que desee a través del formulario de modificación. Posteriormente deberá pulsar la opción de guardar.
Escenario Alternativo:	**1, 2, 3, 4:** El usuario puede abandonar el proceso de modificar películas en cualquier momento. **4:** Si el usuario deja vacío el campo de nombre de la película, o modifica el mismo poniendo uno que ya existe, volverá a aparecer el formulario indicando que el campo introducido no es correcto.

CUOI -05	ELIMINAR PELÍCULA
Nombre:	Eliminar película
Actores:	Usuario.
Objetivo:	Eliminar una película de la base de datos.
Descripción:	Un usuario puede eliminar una película por los motivos que desee.
Precondiciones:	Estar autenticado en el sistema y que la película exista en la base de datos.
Poscondiciones:	Se eliminará de la película de la base de datos.
Escenario:	1. El usuario accede al apartado de películas. 2. El usuario **visualiza la película** que desea borrar. 3. El usuario pulsa la opción borrar de la propia película.
Escenario Alternativo:	**1, 2, 3:** El usuario puede abandonar el proceso de eliminar películas en cualquier momento.

CUOI -06	BUSCAR PELÍCULAS
Nombre:	Buscar película
Actores:	Usuario.
Objetivo:	Buscar una película almacenada en la base de datos.
Descripción:	Un usuario puede buscar las películas que cumplan determinadas condiciones de búsqueda.
Precondiciones:	Estar autenticado en el sistema.
Poscondiciones:	Se actualizará la lista de películas minimizadas con los resultados obtenidos de la consulta realizada.
Escenario:	1. El usuario accede al apartado de películas. 2a. El usuario busca la película a través de las que aparecen en la lista de películas minimizadas. 2b. El usuario puede filtrar películas por determinados campos para tener menos sobre las que buscar en la vista de miniatura.
Escenario Alternativo:	**1, 2a, 2b:** El usuario puede abandonar el proceso de buscar películas en cualquier momento. **2b:** Si el usuario no introduce ningún campo en el formulario de búsqueda, se obtendrán todas las películas almacenadas en la base de datos. **2b:** Si tras buscar una película, no aparece ningún resultado en la vista de miniatura, significará que no hay ninguna de ellas que cumpla con los campos de búsqueda introducidos.

CUOI -07	MODIFICAR CAMPOS VISTA MINIATURA
Nombre:	Modificar campos vista miniatura
Actores:	Usuario.
Objetivo:	Modificar los campos de la lista en miniatura de las películas.
Descripción:	Un usuario puede personalizar los campos que se mostrarán en la lista de miniatura de películas.
Precondiciones:	Estar autenticado en el sistema.
Poscondiciones:	Cada elemento de la lista de miniatura de películas mostrará los campos que se hayan seleccionado.
Escenario:	1. El usuario accede al apartado de películas. 2. El usuario selecciona del menú de búsquedas los campos que desea que se muestren en la lista de resultados de la consulta (lista de miniatura). 3. El usuario pulsará el botón de buscar para actualizar la lista.
Escenario Alternativo:	**1, 2, 3:** El usuario puede abandonar el proceso de modificar los campos de la vista en miniatura en cualquier momento. **2:** Aunque el usuario desmarque todos los campos posibles a mostrar, siempre aparecerá el cartel de la película.

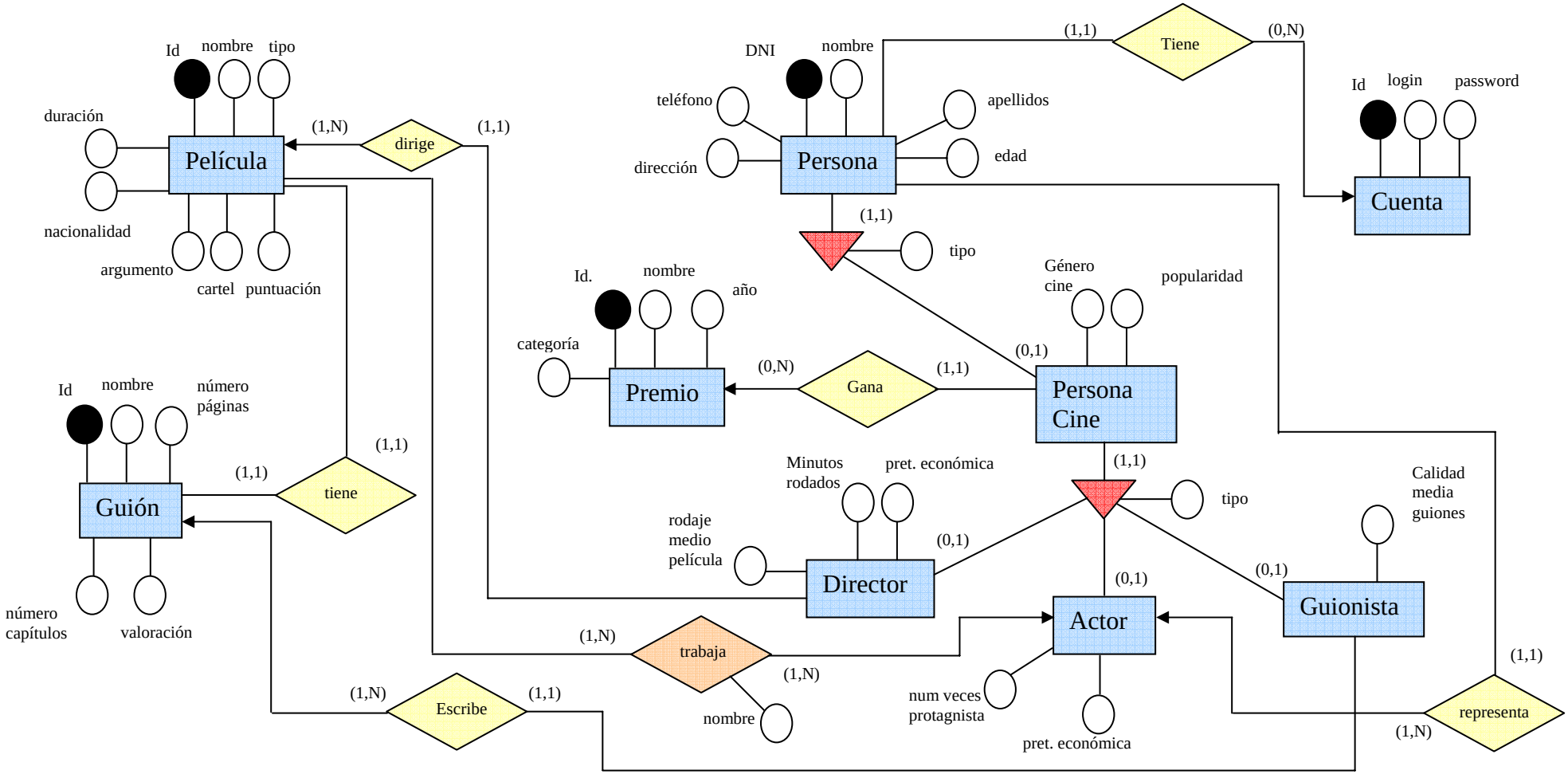
CUOI -08	VISUALIZAR PELÍCULA
Nombre:	Visualizar película
Actores:	Usuario.
Objetivo:	Visualizar una película de la base de datos.
Descripción:	Un usuario puede visualizar todos los campos de una película.
Precondiciones:	Estar autenticado en el sistema y que la película en cuestión haya sido creada.
Poscondiciones:	Se mostrará el contenido de la película seleccionada.
Escenario:	1. El usuario accede al apartado de películas. 2a. El usuario elige la película que desee de las que aparecen en la vista de miniatura. 2b. El usuario puede buscar una película en función a determinados campos, y posteriormente elegir alguna de las que aparecen en la vista de miniatura. 2c. El usuario elige la película de entre las contenidas en la lista del TOP-10. 3. El usuario pulsará sobre la película que desee mostrar.
Escenario Alternativo:	**1, 2a, 2b, 2c, 3:** El usuario puede abandonar el proceso de visualizar una película en cualquier momento. **2b:** Si el usuario introduce filtros que no dan ningún resultado en la búsqueda, la película no podrá visualizarse.

CUOI -09	CERRAR PELÍCULA
Nombre:	Cerrar película
Actores:	Usuario.
Objetivo:	Cerrar una película.
Descripción:	Un usuario puede cerrar la vista maximizada de una película.
Precondiciones:	Estar autenticado en el sistema y que la película en cuestión haya sido visualizada.
Poscondiciones:	Se ocultará la película en cuestión.
Escenario:	1. El usuario accede al apartado de películas. 2. El usuario **visualizará una película** en cuestión. 3. El usuario pulsará sobre la opción cerrar dentro de la propia película.
Escenario Alternativo:	**1, 2, 3:** El usuario puede abandonar el proceso de cerrar una película en cualquier momento.

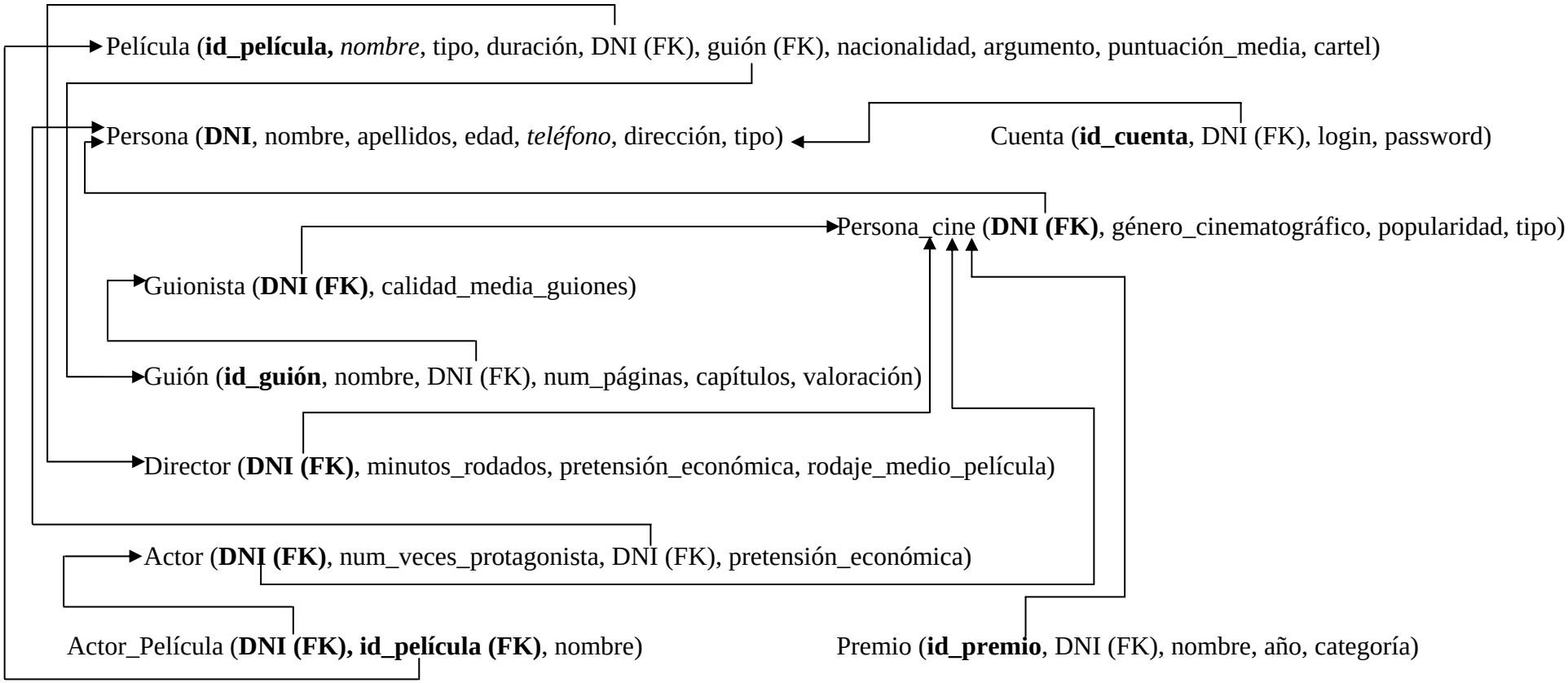
CUOI -10	PUNTUAR PELÍCULA
Nombre:	Puntuar película
Actores:	Usuario.
Objetivo:	Puntuar una película.
Descripción:	Un usuario puede puntuar una película con valores del 1-5.
Precondiciones:	Estar autenticado en el sistema y que la película exista en la base de datos.
Poscondiciones:	Se actualizará la puntuación media teniendo en cuenta la puntuación dada.
Escenario:	1. El usuario accede al apartado de películas. 2. El usuario **visualiza la película** que desee. 3. El usuario pulsará las estrellas que desea dar a la película.
Escenario Alternativo:	**1, 2, 3:** El usuario puede abandonar el proceso de puntuar películas en cualquier momento.

CUOI-11	MODIFICAR VISTA TOP-3
Nombre:	Modificar vista TOP-3
Actores:	Usuario.
Objetivo:	Modificar la vista de la lista TOP-3 de usuarios.
Descripción:	El usuario podrá cambiar los campos que se muestran en la lista contenedora de las tres personas más colaborativas.
Precondiciones:	Estar autenticado en el sistema.
Poscondiciones:	Se actualizará la vista de la lista del TOP-3 de usuarios con los campos que proceda.
Escenario:	1. El usuario deberá acceder a Inicio. 2. El usuario deberá seleccionar sobre la lista otra vista de las que se proporcionen.
Escenario Alternativo:	**1, 2:** El usuario puede abandonar el proceso de modificación de la vista del TOP-3 en cualquier momento.

2.6. Modelo de datos Entidad-Relación



2.7. Modelo de datos Relacional



3. Planificación y Presupuesto de la aplicación

A continuación se expone el presupuesto y planificación necesarios para llevar a cabo la aplicación en cuestión:

3.1. Desglose por fases del proyecto

A continuación se presentan las horas totales de cada una de las fases seguidas en la creación de la aplicación.

Fase\Horas	Horas totales
Análisis	15
Diseño	40
Implementación	100
Documentación	40
Pruebas	105
Total horas	300

Tabla 38: horas/fase aplicación

3.2. *Planificación*

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Creación de la aplicación WPF**	**46 días**	**mié 10/12/08**	**lun 19/01/09**
2	Análisis	3 días	mié 10/12/08	jue 11/12/08
3	Diseño	8 días	jue 11/12/08	jue 18/12/08
4	**Implementación**	**27 días**	**jue 18/12/08**	**lun 12/01/09**
5	Codificación	27 días	jue 18/12/08	lun 12/01/09
6	Pruebas	27 días	jue 18/12/08	lun 12/01/09
7	Documentación	8 días	lun 12/01/09	lun 19/01/09

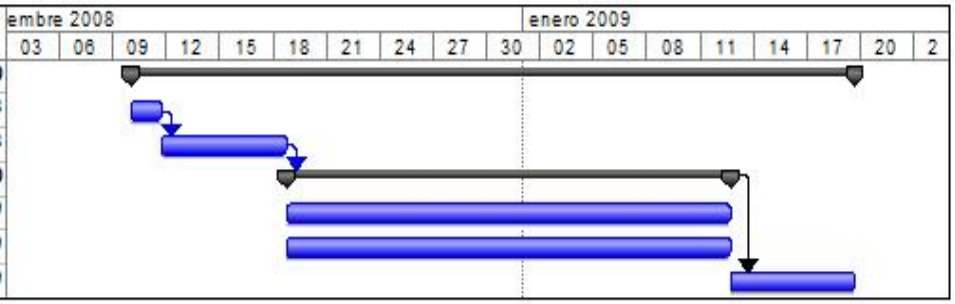


Ilustración 48: Gantt Aplicación

3.3. Salarios por categoría

Para conocer el gasto de la empresa en relación al trabajo de sus empleados o personal que conforma la misma es necesario conocer los salarios de los trabajadores.

Los salarios medios se definen en la siguiente tabla:

Cargo	Sueldo Bruto Año	Sueldo Bruto mes	Coste Hora/empleado
Jefe de Proyecto	60.000 €	5.000 €	31,25 €
Analista	30.000 €	2.500 €	15,625 €
Programador	22.000 €	1.833,33 €	11,45 €

Tabla 39: Coste Salarios puesto/hora aplicación

A continuación, tras conocer los salarios de los empleados hay que verificar cuantas horas del proyecto va a dedicar cada uno de ellos, en relación a las distintas fases del proyecto, incluyendo los costes asociados:

Fase	Jefe de Proyecto (horas)	Analista (horas)	Programador (horas)	Total (€)
Análisis	5	10		312,5
Diseño	10	30		781,25
Implementación		30	70	1270,25
Documentación	20	20		937,5
*Pruebas**		20	15	484,25
TOTAL	**35*31,25= 1.093,75€**	**110*15,625= 1.718,75€**	**85*11,45= 973,25€**	**3.785,75**

Tabla 40: Coste salarios de personal/fase aplicación

* de las 105 horas totales de pruebas mostradas en el desglose por fases del proyecto, se reflejan únicamente 35 puesto que son las pruebas que se han llevado a cabo durante la fases de análisis, diseño e implementación. El resto de horas,

concretamente 70 son las invertidas en las pruebas finales, cuyo desglose se mostrará con mayor detalle en posteriores apartados.

3.4. Gastos de personal imputables al proyecto

A partir de la tabla anterior es posible sacar la relación de costes de los empleados del proyecto en relación a las horas totales que va a dedicar cada uno de ellos al mismo:

Empleado	Horas	Coste
Jefe de Proyecto	35 h	35*31,25=1.093,75€
Analista	110 h	110*15,625=1.718,75€
Programador	85 h	85*11,45=973,25€
TOTAL	**230 h**	**3.785,75 €**

Tabla 41: Coste horas/empleados aplicación

3.5. Gastos en materiales

En este apartado es necesario incluir todo el material necesario para llevar a cabo el proyecto. En principio al tratarse de un proyecto informático, el material necesario estará formado por los equipos de trabajo del personal desarrollador de la empresa y por materiales varios. En la siguiente tabla se resume el total de gastos de materiales:

Material	Cantidad	Coste
Ordenador Portatil Dell Inspiron 64 bits 1 GB RAM 60 GB Disco Duro	1	499 €
Ordenador de sobremesa HP Pavilion serie a6700 4GB RAM 640 GB Disco Duro	2	599 € / unidad
Sistema Operativo Windows Vista	3	300 € / unidad

Profesional		
TOTAL	-	**2.597 €**

Tabla 42: Gastos materiales aplicación

3.6. Resumen del presupuesto

En esta tabla se especifican todos los gastos obtenidos anteriormente, para la presentación del resumen del presupuesto, al cual debe incluirse el IVA:

Gasto	Coste
Empleados	3.785,75 €
Gastos en materiales	2.597 €
SUBTOTAL	6.382,75 €
IVA (16%)	1.021,24 €
TOTAL	**7.403,99 €**

Tabla 43: Resumen presupuesto aplicación

El presupuesto final de la realización de la aplicación *Gestiona* asciende a un TOTAL de **7.403,99 €** (SIETE MIL CUATROCIENTOS TRES, CON NOVENTA Y NUEVE EUROS) **IVA INC.**

V

Pruebas de UI sobre aplicación WPF

V. Pruebas de UI sobre aplicación WPF

1. Introducción sobre la puesta en marcha de la metodología

A continuación se procede a verificar la calidad de la aplicación implementada, en términos de interfaz. Para llevar a cabo dicho proceso de aseguramiento de calidad se va a hacer uso de la metodología definida anteriormente.

Durante la fase de implementación se han llevado a cabo las pertinentes pruebas de caja blanca para verificar que determinados métodos o módulos funcionan correctamente, e incluso para solucionar deficiencias detectadas durante dicha fase. Una vez terminada esta fase, llegando a un punto en el que el producto es estable, este deberá ser liberado para que el equipo de calidad lleve a cabo las pruebas pertinentes sobre el mismo.

En la fase de pruebas se deberá verificar exhaustivamente la calidad del sistema, intentando encontrar el mayor número de errores. Por ello, las mismas deberán ser de distintas tipologías, y deberán abarcar todos los módulos funcionales de la aplicación.

A continuación, se expondrán las pruebas referentes a la interfaz gráfica de usuario, para las cuales se seguirá minuciosamente la metodología definida.

2. Puesta en marcha de la metodología de QA

La metodología que se utilizará diferencia tres partes fundamentalmente, por lo que en este apartado de puesta en marcha existirán tales partes:

- **Estudio de viabilidad:** en este apartado se describirá lo necesario para considerar si la puesta en marcha de la metodología es factible con objeto a la aplicación desarrollada.
- **Extracción de conocimiento y generación de documentación preliminar:** en este apartado se expondrá toda la información posible de la aplicación con motivo de facilitar el trabajo a la hora de crear los diferentes planes de prueba, así como para mejorar la calidad de los mismos.
- **Plan de aseguramiento de calidad:** en este apartado se definirán las diferentes pruebas a validar sobre la aplicación en cuestión, con las cuales se pueda verificar el comportamiento de la aplicación.

2.1. Estudio de viabilidad

El primer paso es el de nombrar una persona encargada de llevar a cabo el estudio de viabilidad. El encargado de llevar a cabo el nombramiento será **Jesús Sánchez Jiménez**, en calidad de Jefe Encargado de Calidad. La persona destinada a tal fin desempeñará el rol de *analista de sistemas* y será **Antonio Gómez López**.

2.1.1. Estudio de las características del sistema

A continuación, el analista de sistemas deberá verificar si la aplicación sobre la que se pretende garantizar la calidad es compatible con la metodología.

Tras el estudio de la aplicación y de la documentación disponible, el documento *Req-Carac-Sist* generado queda como se presenta a continuación:

REQ-CARAC-SIST		
CUMPLIMIENTO REQUISITOS NECESARIOS		
Aplicación WPF	Si [X]	No [ ]
Aplicación Visual	Si [X]	No [ ]
Acceso permitido a la UI mediante programación.	Si [X]	No [ ]
Controles identificables	Si [X]	No [ ]
Máquinas con S.O. Windows XP o Vista	Si [X]	No [ ]
ESTUDIO DE CARACTERÍSTICAS DEL SISTEMA		
Descripción	Está implementada en C# y XAML.	
Observaciones	El equipo de calidad puede no estar familiarizado con estos lenguajes de programación.	

Favorable	Si [ ] No [X]
Descripción	La forma de trabajar en WPF permite que en la codificación de una aplicación se separe la parte lógica de la parte visual. El código en C# expresa la lógica del programa, y el escrito en XAML expresa la parte visual.
Observaciones	Esta característica del sistema es favorable ya que hace más intuitivo el estudio de la aplicación. En la mayor parte de los casos la extracción de conocimiento se hará sobre los ficheros XAML que es donde se crean los controles que aparecen en la interfaz.
Favorable	Si [X] No [ ]
Descripción	La aplicación ha sido desarrollada con el Microsoft .NET Framework 3.5.
Observaciones	Este hecho puede suponer que el personal que trabaje en el proyecto deba tener nociones sobre las características que incorpora el mismo.
Favorable	Si [ ] No [X]

Hay que mencionar que las máquinas de las que se dispone son compatibles con las características enumeradas anteriormente, pudiendo instalar cualquier software necesario en caso de no disponer de él.

2.1.2. Asignación de personal

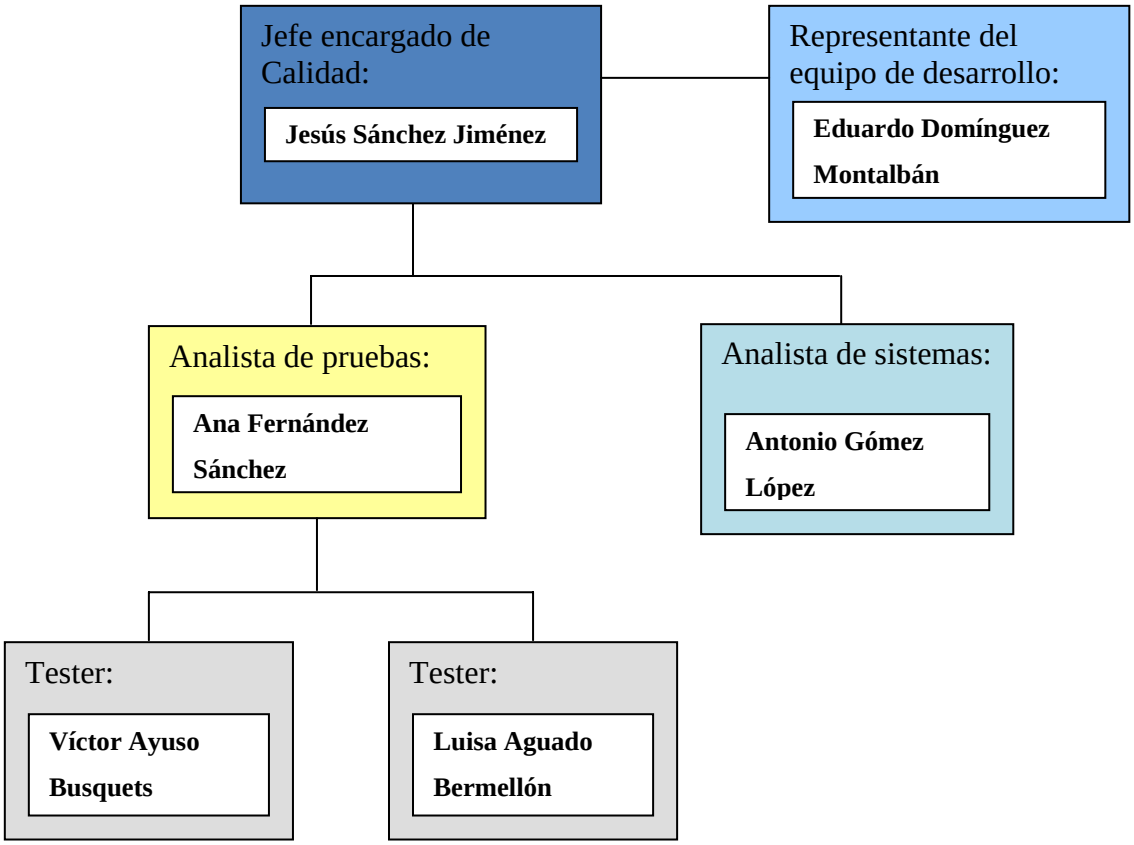
Una vez que se tiene conocimiento de que se cumplen los requisitos necesarios para que la metodología sea compatible con la aplicación objeto de QA, se procederá a la formación de un equipo de calidad por parte del JEC.

Para la elección del personal a los roles que menciona la metodología, se hará uso del documento *Req-Carac-Sist* que se acaba de generar para intentar que el equipo de calidad esté lo mejor preparado posible para llevar a cabo la tarea de aseguramiento de calidad.

EQUIP-CAL	
Jefe encargado de Calidad	Jesús Sánchez Jiménez
Motivo de la elección	
5 años de experiencia como jefe de proyectos de calidad, cumpliendo siempre con los objetivos planteados desde un principio.	
Representante del equipo de desarrollo	Eduardo Domínguez Montalbán
Motivo de la elección	
Se trata de la persona encargada de supervisar el trabajo realizado por parte del equipo de desarrollo. Por ello, sería muy útil emplearlo como mediador entre el propio equipo de desarrollo y el equipo de calidad.	
Analista de sistemas	Antonio Gómez López
Motivo de la elección	

2 años de experiencia como analista de sistemas. Conocimientos de WPF del Microsoft .NET Framework 3.0.	
Analista de pruebas	Ana Fernández Sánchez
Motivo de la elección	
Persona con experiencia más que probada en el análisis y diseño de pruebas.	
Tester	Víctor Ayuso Busquets
Motivo de la elección	
Experiencia en ejecución de planes de pruebas en otros proyectos de la misma empresa.	
Tester	Luisa Aguado Bermellón
Motivo de la elección	
Experiencia en ejecución de planes de pruebas en otros proyectos de la misma empresa.	

Una vez que se han asignado las diferentes personas a cada rol establecido, se procederá a establecer la jerarquía de responsabilidades del propio equipo de calidad:



La metodología utilizada menciona como tarea opcional la creación de un equipo suplente. En este caso se considera útil esta tarea, desarrollando el siguiente documento:

EQUIP-CAL-SUPL	
Jefe encargado de Calidad	Lidia Aranguren Zubizarreta
Motivo de la elección	
1 año de experiencia en dirección proyectos de calidad.	
Analista de sistemas	**Francisco Chacón Herreros**
Motivo de la elección	
Experiencia como analista de sistemas.	
Analista de pruebas	Ramón Serrano Hervás
Motivo de la elección	
Ya ha trabajado en otros proyectos de este tipo en otras empresas.	
Tester	Sandra Herranz Crespo
Motivo de la elección	
Formación relativa a la puesta en marcha de planes de pruebas.	

2.1.3. Aceptación o rechazo del estudio de viabilidad

Llegados a este punto el JEC en compañía del analista de sistemas deciden sobre la viabilidad final del proyecto en función a los datos recopilados en el documento Req-Carac-Sist.

Debido a que se ha llegado a esta tarea, implícitamente se han cumplido los requisitos necesarios para que la metodología sea compatible. Sin embargo habrá que estudiar si las demás características que aparecen en el documento Req-Carac-Sist, concretamente las que no tienen el campo favorable activado, son determinantes para denegar la viabilidad del proyecto.

La decisión sobre este tipo de características es subjetiva por lo que los roles encargados de ello van a proceder a exponer en las observaciones los problemas que se presuponían, concluyendo con su opinión al respecto.

Aprob-Viabilidad	
Observaciones pertinentes	
Observación:	"Puede que el equipo de calidad no esté familiarizado con el lenguaje en el que está implementada la aplicación." Este hecho no se considera un problema determinante en nuestro caso puesto que Antonio Gómez López, analista de sistemas, sí tiene conocimientos y experiencia con esta tecnología. Los demás roles no necesitan de estos conocimientos debido a que la labor que se desempeña en cada uno de ellos no lo requiere.
Observación:	"Sería necesario que el analista de sistemas tuviera conocimientos del Microsoft .NET Framework 3.5." Esta situación no se cumple, sin embargo, dicha persona es conocedora del Framework 3.0, por lo que no le supondrá prácticamente ningún esfuerzo aprender el 3.5.
Motivos por los que se aprueba/deniega la viabilidad del proyecto	
Ninguna de las características de la aplicación hace que la viabilidad del proyecto se vea perjudicada. Además se dispone tanto de personal cómo de medios técnicos y materiales para poder llevar a cabo dicho proyecto.	

Aprobado [X]		**Denegado** [ ]
Analista de sistemas	Antonio Gómez López	Firma:
Jefe encargado de calidad	Jesús Sánchez Jiménez	Firma:
	Fecha 09-01-2009	

2.2. Extracción de conocimiento y generación de documentación preliminar

Como consecuencia de la resolución positiva del análisis de viabilidad, el proyecto de calidad continuará. El siguiente paso a llevar a cabo, según la metodología, será el de extraer el mayor conocimiento posible sobre la aplicación objeto de QA.

2.2.1. Extracción de módulos funcionales

El analista de sistemas será el encargado de llevar a cabo esta tarea. En ella, se tomará un primer contacto de la funcionalidad que encierra la aplicación.

MÓDULOS-FUNCIONALES	
GESTIÓN DE USUARIOS	
NOMBRE	Gestión de usuarios
DESCRIPCIÓN	El citado módulo funcional engloba la funcionalidad relacionada con la gestión de los usuarios.
FUNCIONALIDADES ASOCIADAS	
Acceder a la aplicación: Un usuario accede a la aplicación autenticándose en la misma.	
Salir de la aplicación: Un usuario que previamente se ha autenticado, cierra su sesión.	

GESTIÓN DE PELÍCULAS	
NOMBRE	Gestión de películas
DESCRIPCIÓN	El citado módulo funcional engloba toda la funcionalidad relacionada con la gestión de películas.
FUNCIONALIDADES ASOCIADAS	
Añadir película: Un usuario añade una película en la base de datos.	
Modificar película: Un usuario modifica cualquier campo de una película.	
Eliminar película: Un usuario elimina una película de la base de datos.	
Buscar película: Un usuario busca una película en función a determinados filtros de búsqueda.	
Puntuar película: Un usuario puntúa una película.	
Visualizar película: Un usuario visualizará todos los campos pertenecientes a una película.	
Cerrar película: Un usuario cerrará una película que previamente ha visualizado.	
Visualizar TOP 10 de películas: Un usuario visualizará una lista con el TOP 10 de las películas.	
GESTIÓN DE VISUALIZACIÓN	
NOMBRE	Gestión de visualización
DESCRIPCIÓN	El citado módulo funcional engloba la funcionalidad referente a la configuración que proporciona la aplicación para la visualización personalizada de campos por parte del usuario.
FUNCIONALIDADES ASOCIADAS	
Campos personalizables en la lista de miniatura de películas: Un usuario personaliza los campos que desea que aparezcan en la lista minimizada de películas.	

Diferentes vistas de lista del TOP 3 de usuarios: Un usuario selecciona la vista que desee en función a si desea obtener más o menos información.
Diferentes vistas de lista del TOP 10 de películas: Un usuario selecciona la vista que desee en función a si desea obtener más o menos información.

2.2.2. Obtención de casos de uso candidatos

Durante esta etapa, el analista de pruebas ha revisado los casos de uso existentes, concluyendo que no será necesario pedir al equipo de desarrollo que se cree ninguno más.

A continuación se muestra el documento que relaciona la funcionalidad que se considera importante para ser probada en un futuro con sus correspondientes casos de uso:

CASOS-USO-CANDIDATOS	
Nombre funcionalidad:	Acceder a la aplicación
Id. del caso de uso:	CUOA-01
Nombre funcionalidad:	Salir de la aplicación
Id. del caso de uso:	CUOA-02
Nombre funcionalidad:	Añadir película
Id. del caso de uso:	CUOI-03
Nombre funcionalidad:	Modificar película
ID. DEL CASO DE USO:	CUOI -04
Nombre funcionalidad:	Eliminar película
ID. DEL CASO DE USO:	CUOI -05
Nombre funcionalidad:	Buscar películas
ID. DEL CASO DE USO:	CUOI -06
Nombre funcionalidad:	Puntuar película
ID. DEL CASO DE USO:	CUOI -10

Nombre funcionalidad:	Visualizar película
ID. DEL CASO DE USO:	CUOI -08
Nombre funcionalidad:	Cerrar película
ID. DEL CASO DE USO:	CUOI -09
Nombre funcionalidad:	Visualizar TOP 10 de películas
ID. DEL CASO DE USO:	CUOI-01
Nombre funcionalidad:	Campos personalizables en la lista de miniatura de películas
ID. DEL CASO DE USO:	CUOI -07
Nombre funcionalidad:	Diferentes vistas de lista del TOP 3 de usuarios
ID. DEL CASO DE USO:	CUOI-11
Nombre funcionalidad:	Diferentes vistas de lista del TOP 10 de películas:
ID. DEL CASO DE USO:	CUOI-02

2.2.3. Casos de uso objeto de QA

Del conjunto de funcionalidades obtenidas en el documento anteriormente expuesto (CASOS-USO-CANDIDATOS), se procederá a elegir un subconjunto del mismo, el cual será sobre el que nos centraremos para verificar la calidad de la aplicación. Dicho subconjunto de funcionalidades/casos de uso será seleccionado por parte del analista de pruebas, y supervisado por el JEC:

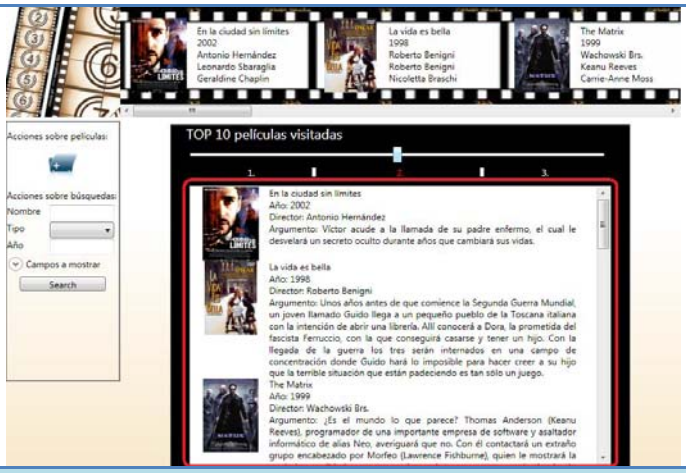
CASOS-USO-OBJETO	
MÓDULO FUNCIONAL	**Gestión de Visualización**
CASO DE USO	CUOI-02
ESTADO	Finalizado
MÓDULO FUNCIONAL	**Gestión de películas**
CASO DE USO	CUOI-03
ESTADO	Finalizado
MÓDULO	**Gestión de películas**

FUNCIONAL	
CASO DE USO	CUOI-04
ESTADO	En proceso
MÓDULO FUNCIONAL	**Gestión de películas**
CASO DE USO	CUOI-05
ESTADO	En proceso
MÓDULO FUNCIONAL	**Gestión de películas**
CASO DE USO	CUOI-06
ESTADO	Finalizado
MÓDULO FUNCIONAL	**Gestión de Visualización**
CASO DE USO	CUOI-07
ESTADO	Finalizado
MÓDULO FUNCIONAL	**Gestión de películas**
CASO DE USO	CUOI-10
ESTADO	Finalizado

2.2.4. Controles por caso de uso

Una vez conocidos cada uno de los casos de uso que serán objeto de QA, los *Tester* (Víctor Ayuso Busquets y Luisa Aguado Bermellón) deberán proceder a estudiar los controles que intervienen en cada uno de ellos para disponer de la mayor información posible:

CONTROLES-CASOS-USO	
CUOI-02	
IDENTIFICADOR	**CUOI-02-1**
TIPO CONTROL	Slider
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control se utiliza pulsando sobre el puntero y desplazándolo a través de la barra, para finalmente soltarlo en el lugar que se pretenda. - Sirve para que se muestre una u otra vista de la lista del TOP-10 de películas en función al lugar en el que se mueva el puntero. Hay tres posibles vistas para la lista.
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-02-2**
TIPO CONTROL	ListView
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control se utiliza para mostrar contenido en forma de lista. - Esta lista muestra determinados campos en función a la vista que se esté mostrando en cada momento: o Vista 1: nombre película y argumento. o Vista 2: nombre película, año, director y argumento.

	o Vista 3: nombre película, año, director, guionista y argumento.
IMAGEN DEL CONTROL	
CUOI-03	
IDENTIFICADOR	**CUOI-03-1**
TIPO CONTROL	CustomImage
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Sirve para que al pinchar en la misma, aparezca un formulario para crear la nueva película.
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-03-2**
TIPO CONTROL	TextBox
DESCRIPCIÓN CONTROL	- Aparecen en el formulario de creación de una película. - Este control permite introducir texto. - Hay nueve TextBox que representan los siguientes campos de una película: o Sinopsis.

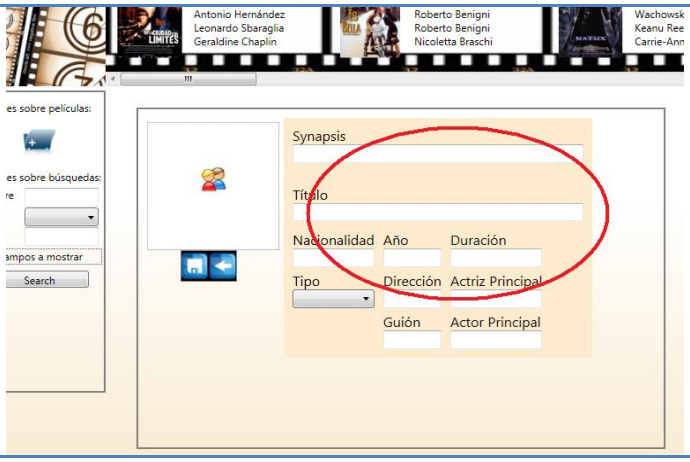
	o Título. o Nacionalidad. o Año. o Duración. o Dirección. o Guión. o Actor principal. o Actriz principal.
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-03-3**
TIPO CONTROL	ComboBox
DESCRIPCIÓN CONTROL	- Aparece en el formulario de creación de una película. - Este control permite elegir alguno de los ítems que contiene. - Este ComboBox representa el tipo de una película, la cual puede tomar los siguientes valores: o Acción. o Comedia. o C. Ficción. o Drama. o Suspense. o Terror.

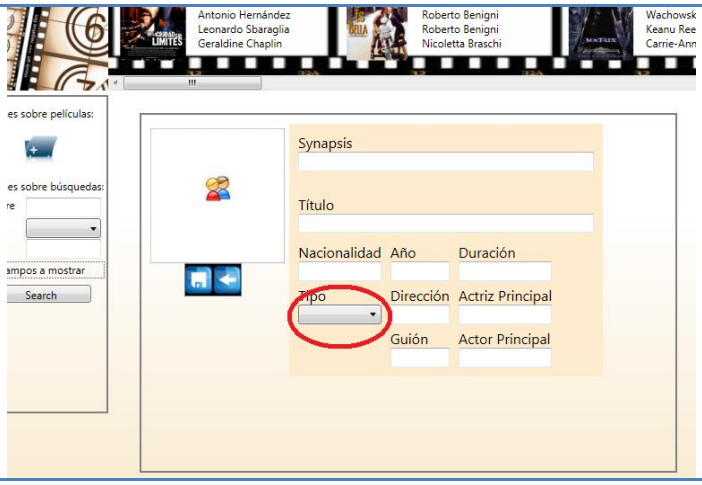
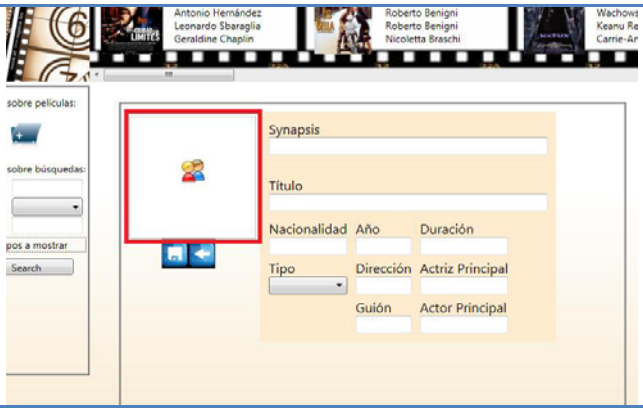
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-03-4**
TIPO CONTROL	Image
DESCRIPCIÓN CONTROL	- Aparece en el formulario de creación de una película. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada, abriéndose una ventana que permitirá elegir un fichero. - Esta imagen es la que se guardará como cartel de la película.
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-03-5**
TIPO CONTROL	M4Image
DESCRIPCIÓN CONTROL	- Aparece en el formulario de creación de una película. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Sirve para que al pinchar en la misma, se guarden los datos introducidos por el usuario en lo que a creación de una película se refiere.

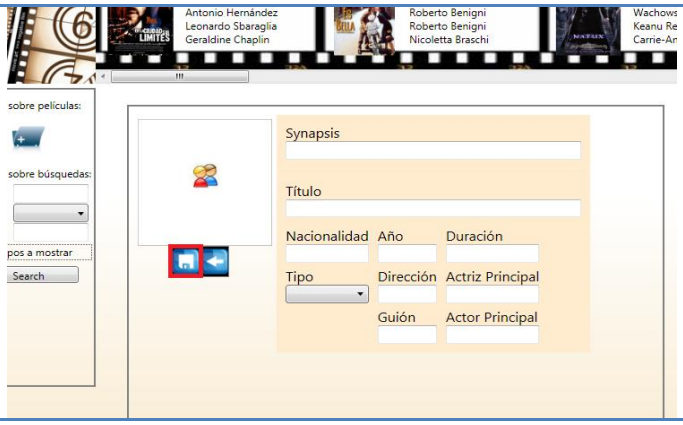
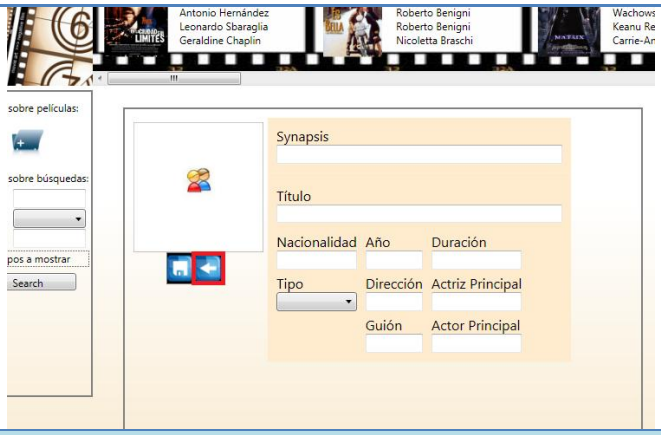
IMAGEN DEL CONTROL	
Identificador	**CUOI-03-6**
TIPO CONTROL	M4Image
DESCRIPCIÓN CONTROL	- Aparece en el formulario de creación de una película. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Sirve para que al pinchar en la misma, se cierre el formulario de creación de película sin guardar.
IMAGEN DEL CONTROL	
CUOI-04	
IDENTIFICADOR	**CUOI-04-1**
TIPO CONTROL	M4Image
DESCRIPCIÓN CONTROL	- Aparece en la vista ampliada de una película. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Sirve para que al pinchar en la misma, aparezca un formulario para modificar los campos de una película.

IMAGEN DEL CONTROL.	
IDENTIFICADOR	**CUOI-04-2**
TIPO CONTROL	TextBox
DESCRIPCIÓN CONTROL	- Aparecen en el formulario de modificación de una película. - Este control permite introducir texto. - Hay nueve TextBox que representan los siguientes campos de una película: o Sinopsis. o Título. o Nacionalidad. o Año. o Duración. o Dirección. o Guión. o Actor principal. o Actriz principal.
IMAGEN DEL CONTROL	Misma imagen que CUOI-03-2, pero cada TextBox tendrá el valor correspondiente de la película seleccionada.
IDENTIFICADOR	**CUOI-04-3**
TIPO CONTROL	ComboBox
DESCRIPCIÓN CONTROL	- Aparece en el formulario de modificación de una película. - Este control permite elegir alguno de los ítems que contiene. - Este ComboBox representa el tipo de una película, la cual puede tomar los siguientes valores: o Acción. o Comedia. o C. Ficción.

	o Drama. o Suspense. o Terror.
IMAGEN DEL CONTROL	Misma imagen que CUOI-03-3, pero el ComboBox tendrá el valor de la película seleccionada.
IDENTIFICADOR	**CUOI-04-4**
TIPO CONTROL	Image
DESCRIPCIÓN CONTROL	- Aparece en el formulario de modificación de una película. - Esta imagen es la que se guardará como cartel de la película.
IMAGEN DEL CONTROL	Misma imagen que CUOI-03-4, pero la imagen tendrá el cartel de la película seleccionada.
IDENTIFICADOR	**CUOI-04-5**
TIPO CONTROL	M4Image
DESCRIPCIÓN CONTROL	- Aparece en el formulario de modificación de una película. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Sirve para que al pinchar en la misma, se guarden los datos introducidos por el usuario en lo que a modificación de una película se refiere.
IMAGEN DEL CONTROL	Misma imagen que CUOI-03-5.
IDENTIFICADOR	**CUOI-04-6**
TIPO CONTROL	M4Image
DESCRIPCIÓN CONTROL	- Aparece en el formulario de modificación de una película. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Sirve para que al pinchar en la misma, se cierre el formulario de modificación de película sin guardar.
IMAGEN DEL CONTROL	Misma imagen que CUOI-03-6.
CUOI-05	
IDENTIFICADOR	**CUOI-05-1**

TIPO CONTROL	M4Image
DESCRIPCIÓN CONTROL	- Aparece en la vista ampliada de una película. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Sirve para que al pinchar en la misma, se borre de la base de datos una película.
IMAGEN DEL CONTROL	
CUOI-06	
IDENTIFICADOR	**CUOI-06-1**
TIPO CONTROL	PeliculaMinimizada
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Se trata de un contenedor de controles con una imagen y un número variable de TextBlock. - Hay tantos controles de este tipo como películas haya en la base de datos. - Sirve para obtener información resumida de una película. Además al pinchar en el mismo, se visualizará de forma maximizada la información relativa a una película.
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-06-2**
TIPO CONTROL	TextBox

DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control permite introducir texto. - Hay dos TextBox que representan los filtros de búsqueda que se utilizarán: o Nombre. o Año.
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-06-3**
TIPO CONTROL	ComboBox
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control permite elegir alguno de los ítems que contiene. - Este ComboBox representa el filtro de búsqueda por tipo de película, la cual puede ser: o Acción. o Comedia. o C. Ficción. o Drama. o Suspense. o Terror.

IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-06-4**
TIPO CONTROL	Button
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control permite ser invocado. - La invocación de dicho control provocará que se lleve a cabo la búsqueda, en función a los filtros que se pueden haber rellenado anteriormente.
IMAGEN DEL CONTROL	
CUOI-07	
IDENTIFICADOR	**CUOI-07-1**
TIPO CONTROL	PeliculaMinimizada
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Se trata de un contenedor de controles con una imagen y un número variable de TextBlock. - Hay tantos controles de este tipo como películas haya en la base de datos. - Sirve para obtener información resumida de una película. Además al pinchar en el mismo, se visualizará

	de forma maximizada la información relativa a una película.
IMAGEN DEL CONTROL	Misma imagen que CUOI-06-1.
IDENTIFICADOR	**CUOI-07-2**
TIPO CONTROL	Expander
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control permite colapsar o expandir su contenido.
IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-07-3**
TIPO CONTROL	CheckBox
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control permite ser marcado o desmarcado. - Hay cuatro controles de este tipo, los cuales harán visibles los siguientes campos de los controles *PeliculaMinimizada*: o Nombre de la película. o Año de la película. o Director. o Actores principales.

IMAGEN DEL CONTROL	
IDENTIFICADOR	**CUOI-07-4**
TIPO CONTROL	Button
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Este control permite ser invocado. - La invocación de dicho control provocará que se actualice la vista de la lista de películas minimizadas.
IMAGEN DEL CONTROL	Misma imagen que CUOI-06-4.
	CUOI-10
IDENTIFICADOR	**CUOI-10-1**
TIPO CONTROL	Image
DESCRIPCIÓN CONTROL	- Aparece en el apartado *películas* del menú principal. - Esta imagen tiene la funcionalidad de un botón puesto que puede ser invocada. - Hay cinco controles de este tipo. - Sirve para que al pinchar en la misma, se puntúen las películas.

IMAGEN DEL CONTROL	

2.2.5. Estudio de propiedades y eventos

La última tarea del apartado de extracción de conocimiento trata de obtener la mayor información posible por cada control de los extraídos anteriormente. La tarea será llevada a cabo por el analista de sistemas, y la documentación generada se expone a continuación:

Detalle-Controles		
Detalle de controles de la aplicación objeto de QA		
Identificador:	**CUOI-02-1**	
Tipo control:	Slider	
Propiedades:	Nombre	Valor
	AutomationId	slider1
	Descripción	
	Identificador que servirá a la librería de automatización para encontrar un determinado control.	
	Nombre	Valor
	Width	550
	Descripción	

<table>
<tr><td></td><td colspan="2">Esta propiedad representa la anchura del control.</td></tr>
<tr><td rowspan="11">Eventos:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td>MouseWheel</td><td>actualizarValor</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Evento de ratón que se ejecuta cuando se mueve la rueda del mismo al estar seleccionado el control en cuestión.

El evento mueve el puntero del slider hasta la posición que se desee, actualizando su valor. Si el puntero mencionado pasa de determinado valor, la lista explicada a continuación cambiará de vista. Además se deberá poner en rojo el número que referencia la vista que se está visualizando.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>PreviewMouseLeftButtonUp</td><td>actualizarValor</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Evento de ratón que se ejecuta cuando se levanta el botón izquierdo del ratón después de haber pulsado sobre el puntero del propio control.

El evento mueve el puntero del slider hasta la posición que se desee, actualizando su valor. Si el puntero mencionado pasa de determinado valor, la lista explicada a continuación cambiará de vista.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-02-2</td></tr>
<tr><td>Tipo control:</td><td colspan="2">ListView</td></tr>
<tr><td rowspan="3">Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td>AutomationId</td><td>listaVistos2</td></tr>
<tr><td colspan="2">Descripción</td></tr>
</table>

<table>
<tr><td></td><td>Identificador que servirá a la librería de automatización para encontrar un determinado control.
<table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>Heigth</td><td>380</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Esta propiedad representa la altura del control.</td></tr><tr><td>Nombre</td><td>Valor</td></tr><tr><td>Width</td><td>550</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Esta propiedad representa la anchura del control.</td></tr></table></td></tr>
<tr><td>Identificador:</td><td>CUOI-03-1</td></tr>
<tr><td>Tipo control:</td><td>CustomImage</td></tr>
<tr><td>Propiedades:</td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>AutomationId</td><td>insertarImagen</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Identificador que servirá a la librería de automatización para encontrar un determinado control.</td></tr><tr><td>Nombre</td><td>Valor</td></tr><tr><td>Heigth</td><td>45</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Esta propiedad representa la altura de la imagen.</td></tr><tr><td>Nombre</td><td>Valor</td></tr><tr><td>Width</td><td>45</td></tr><tr><td colspan="2">Descripción</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>Esta propiedad representa la anchura de la imagen.</td></tr>
<tr><td></td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>BorderBrush</td><td>Blue</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Esta propiedad representa el color del borde que recubre a la imagen.</td></tr></table></td></tr>
<tr><td>Eventos:</td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>MouseEnter</td><td>mouseEnterBorder</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Evento de ratón que se ejecuta cuando el mismo entra en la región de la imagen.

El evento crea un borde alrededor de la imagen para saber que esta puede ser pulsada.</td></tr></table></td></tr>
<tr><td></td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>MouseLeave</td><td>mouseLeaveBorder</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Evento de ratón que se ejecuta cuando el mismo sale de la región de la imagen.

El evento destruye el borde que envuelve la imagen.</td></tr></table></td></tr>
<tr><td></td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>Mouse.MouseDown</td><td>crearPelicula</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Evento de ratón que se ejecuta cuando se pulsa el botón izquierdo sobre la imagen.

El evento abre un nuevo formulario para la creación de una</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td colspan="2">película.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-03-2</td></tr>
<tr><td>Tipo control:</td><td colspan="2">TextBlock</td></tr>
<tr><td>Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>AutomationId</td><td>o modificarArgumento
o modificarTitulo
o modificarNacionalidad
o modificarAnyo
o modificarDuracion
o modificarDirector
o modificarActorFem
o modificarActorMasc
o modificarGuionista</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Identificador de los sucesivos controles de tipo TextBlock que servirán a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>MaxWidth</td><td>350</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Esta propiedad representa la anchura máxima que puede tener el control.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>TextWrapping</td><td>Wrap</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Esta propiedad indica que si el texto contenido en el TextBlock llega a la anchura máxima permitida, este deberá truncarse y continuar una línea más abajo.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-03-3</td></tr>
</table>

<table>
<tr><td>Tipo control:</td><td colspan="2">ComboBox</td></tr>
<tr><td rowspan="15">Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td>AutomationId</td><td>modificarTipo</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Identificador que servirá a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>Childs</td><td>o Acción
o Comedia
o C.Ficción
o Drama
o Suspense
o Terror
o “”</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad representa los elementos que contendrá la ComboBox.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>ExpandCollapse</td><td>Collapsed</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad indica si la ComboBox está o no expandida.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>CanSelectMultiple</td><td>False</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad indica si existe la posibilidad de seleccionar más de un elemento hijo a la vez.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-03-4</td></tr>
</table>

<table>
<tr><td>Tipo control:</td><td colspan="2">M4Image</td></tr>
<tr><td rowspan="19">Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td>AutomationId</td><td>modificarImagen</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Identificador que servirá a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>Heigth</td><td>175</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad representa la altura de la imagen.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>Width</td><td>175</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad representa la anchura de la imagen.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>AllowDrop</td><td>True</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad indica que se permite soltar elementos sobre la región de la imagen.</td></tr>
<tr><td rowspan="7">Eventos:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td>Drop</td><td>sueltaImagen</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Evento de ratón que se ejecuta cuando se suelta algún elemento sobre la región que ocupa la imagen en cuestión.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>Mouse.MouseDown</td><td>abrirVentanaFicheros</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Evento de ratón que se ejecuta cuando se pulsa el botón izquierdo sobre la imagen.

El evento abre un explorador para seleccionar un determinado elemento.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-03-5</td></tr>
<tr><td>Tipo control:</td><td colspan="2">M4Image</td></tr>
<tr><td>Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>AutomationId</td><td>guardar</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Identificador que servirá a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Heigth</td><td>30</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Esta propiedad representa la altura de la imagen.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Width</td><td>30</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Esta propiedad representa la anchura de la imagen.</td></tr>
<tr><td>Eventos:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Mouse.MouseDown</td><td>guardarCambiosPelicula</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Evento de ratón que se ejecuta después de pulsar sobre la imagen en cuestión.</td></tr>
</table>

Identificador:	**CUOI-03-6**	
Tipo control:	M4Image	
Propiedades:	Nombre	Valor
	AutomationId	volver
	Descripción	
	Identificador que servirá a la librería de automatización para encontrar un determinado control.	
	Nombre	Valor
	Heigth	30
	Descripción	
	Esta propiedad representa la altura de la imagen.	
	Nombre	Valor
	Width	30
	Descripción	
	Esta propiedad representa la anchura de la imagen.	
Eventos:	Nombre	Valor
	Mouse.MouseDown	volverPelicula
	Descripción	
	Evento de ratón que se ejecuta después de pulsar sobre la imagen en cuestión.	
Identificador:	**CUOI-06-1**	
Tipo control:	PeliculaMinimizada	
Propiedades:	Nombre	Valor
	AutomationId	Cada control de este tipo tendrá un identificador: Border_*nombrePelicula*
	Descripción	

<table>
<tr><td></td><td>
Identificador que servirá a la librería de automatización para encontrar un determinado control.
<table>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>Heigth</td><td>150</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad representa la altura del control.</td></tr>
</table>
<table>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>Width</td><td>250</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad representa la anchura del control.</td></tr>
</table>
<table>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>BorderBrush</td><td>Red</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad representa el color del borde que recubre al control.</td></tr>
</table>
</td></tr>
<tr><td>Eventos:</td><td>
<table>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>MouseEnter</td><td>mouseEnterBorder</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Evento de ratón que se ejecuta cuando el mismo entra en la región del control.

El evento crea un borde alrededor del control para saber que este puede ser pulsado.</td></tr>
</table>
<table>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>MouseLeave</td><td>mouseLeaveBorder</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Evento de ratón que se ejecuta cuando el mismo sale de la región</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">del control.

El evento destruye el borde que envuelve al control.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>MouseLeftButtonUp</td><td>pinchaEnPeli</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Evento de ratón que se ejecuta cuando se pulsa el botón izquierdo sobre el control.

El evento abre un nuevo panel, en el cual se expondrá información completa de la película pulsada.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-06-2</td></tr>
<tr><td>Tipo control:</td><td colspan="2">TextBlock</td></tr>
<tr><td>Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>AutomationId</td><td>o textBoxNombre
o textBoxAnyo</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Identificador de los sucesivos controles de tipo TextBlock que servirán a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-06-3</td></tr>
<tr><td>Tipo control:</td><td colspan="2">ComboBox</td></tr>
<tr><td>Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>AutomationId</td><td>comboBoxTipo</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Identificador que servirá a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Childs</td><td>o Acción
o Comedia</td></tr>
</table>

		o C.Ficción o Drama o Suspense o Terror o “”
	Descripción	
	Esta propiedad representa los elementos que contendrá la ComboBox.	
	Nombre	Valor
	ExpandCollapse	Collapsed
	Descripción	
	Esta propiedad indica si la ComboBox está o no expandida.	
	Nombre	Valor
	CanSelectMultiple	False
	Descripción	
	Esta propiedad indica si existe la posibilidad de seleccionar más de un elemento hijo a la vez.	
Identificador:	**CUOI-06-4**	
Tipo control:	Button	
Propiedades:	Nombre	Valor
	AutomationId	botonBuscar
	Descripción	
	Identificador que servirá a la librería de automatización para encontrar un determinado control.	
	Nombre	Valor
	Heigth	20
	Descripción	

<table>
<tr><td></td><td colspan="2">Esta propiedad representa la altura del botón.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Width</td><td>110</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Esta propiedad representa la anchura del control.</td></tr>
<tr><td>Eventos:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Click</td><td>buscarPelicula</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Evento de ratón que se ejecuta cuando se pulsa sobre el control. Este evento produce que se lleve a cabo la búsqueda, actualizando la lista minimizada de películas.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-07-1</td></tr>
<tr><td>Tipo control:</td><td colspan="2">PeliculaMinimizada</td></tr>
<tr><td>Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>AutomationId</td><td>Cada control de este tipo tendrá un identificador: Border_nombrePelicula</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Identificador que servirá a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Heigth</td><td>150</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
<tr><td></td><td colspan="2">Esta propiedad representa la altura del control.</td></tr>
<tr><td></td><td>Nombre</td><td>Valor</td></tr>
<tr><td></td><td>Width</td><td>250</td></tr>
<tr><td></td><td colspan="2">Descripción</td></tr>
</table>

<table>
<tr><td></td><td>Esta propiedad representa la anchura del control.</td></tr>
<tr><td></td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>BorderBrush</td><td>Red</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Esta propiedad representa el color del borde que recubre al control.</td></tr></table></td></tr>
<tr><td>Eventos:</td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>MouseEnter</td><td>mouseEnterBorder</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Evento de ratón que se ejecuta cuando el mismo entra en la región del control.

El evento crea un borde alrededor del control para saber que este puede ser pulsado.</td></tr></table></td></tr>
<tr><td></td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>MouseLeave</td><td>mouseLeaveBorder</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Evento de ratón que se ejecuta cuando el mismo sale de la región del control.

El evento destruye el borde que envuelve al control.</td></tr></table></td></tr>
<tr><td></td><td><table><tr><td>Nombre</td><td>Valor</td></tr><tr><td>MouseLeftButtonUp</td><td>pinchaEnPeli</td></tr><tr><td colspan="2">Descripción</td></tr><tr><td colspan="2">Evento de ratón que se ejecuta cuando se pulsa el botón izquierdo sobre el control.

El evento abre un nuevo panel, en el cual se expondrá información completa de la película pulsada.</td></tr></table></td></tr>
</table>

<table>
<tr><td>Identificador:</td><td colspan="2">CUOI-07-2</td></tr>
<tr><td>Tipo control:</td><td colspan="2">Expander</td></tr>
<tr><td rowspan="11">Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td>AutomationId</td><td>expanderMostrar</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Identificador que servirá a la librería de automatización para encontrar un determinado control.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>Childs</td><td>o Nombre de la película
o Año de la película
o Director
o Actores principales</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad representa los controles contenidos en el control, siendo en este caso de tipo CheckBox.</td></tr>
<tr><td>Nombre</td><td>Valor</td></tr>
<tr><td>ExpandCollapseState</td><td>Collapsed</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad indica que el control puede expandirse, estando por defecto colapsado.</td></tr>
<tr><td>Identificador:</td><td colspan="2">CUOI-07-3</td></tr>
<tr><td>Tipo control:</td><td colspan="2">CheckBox</td></tr>
<tr><td rowspan="2">Propiedades:</td><td>Nombre</td><td>Valor</td></tr>
<tr><td>AutomationId</td><td>Cada control tendrá un id diferente:
- checkNombre
- checkAnyo
- checkDirector</td></tr>
</table>

		- checkActores
	Descripción	
	Identificador que servirá a la librería de automatización para encontrar un determinado control.	
	Nombre	Valor
	IsChecked	True
	Descripción	
	Esta propiedad representa si un control de este tipo está activado o desactivado.	
Eventos:	Nombre	Valor
	Click	modificarVisibilidad
	Descripción	
	Evento de ratón que se ejecutará cuando pulsemos sobre el control. Dicho evento hará que se muestren u oculten los correspondientes campos de la lista de miniatura que se marquen o desmarquen.	
Identificador:	**CUOI-07-4**	
Tipo control:	Button	
Propiedades:	Nombre	Valor
	AutomationId	botonBuscar
	Descripción	
	Identificador que servirá a la librería de automatización para encontrar un determinado control.	
	Nombre	Valor
	Heigth	20
	Descripción	
	Esta propiedad representa la altura del botón.	

	Nombre	Valor
	Width	110
	Descripción	
	Esta propiedad representa la anchura del control.	
Eventos:	Nombre	Valor
	Click	buscarPelicula
	Descripción	
	Evento de ratón que se ejecuta cuando se pulsa sobre el control. Este evento produce que se lleve a cabo la búsqueda, actualizando la lista minimizada de películas.	
Identificador:	**CUOI-10-1**	
Tipo control:	Image	
Propiedades:	Nombre	Valor
	AutomationId	Habrá 5 imágenes, cada una de las cuales tendrá el identificador: star_*numeroSecuencial*
	Descripción	
	Identificador que servirá a la librería de automatización para encontrar un determinado control.	
	Nombre	Valor
	Heigth	25
	Descripción	
	Esta propiedad representa la altura de la imagen.	
	Nombre	Valor
	Width	25
	Descripción	
	Esta propiedad representa la anchura de la imagen.	
Eventos:	Nombre	Valor

	MouseEnter	rellenarEstrellas
	Descripción	
	Evento de ratón que se ejecuta cuando este pasa por encima del control en cuestión. Este evento coloreará las estrellas hasta la que se está apuntando actualmente.	
	Nombre	Valor
	MouseLeave	vaciarEstrellas
	Descripción	
	Evento de ratón que se ejecuta cuando el mismo sale de la región del control. El evento vacía las estrellas dejándolas como estaban antes de haber entrado en la región.	
	Nombre	Valor
	MouseLeftButtonDown	votarPelicula
	Descripción	
	Evento de ratón que se ejecuta cuando se pulsa el botón izquierdo sobre la imagen. El evento modifica la puntuación media de una película en función a la estrella que se haya pulsado.	

2.3. Plan de aseguramiento de calidad

Una vez generada toda la documentación pertinente acerca de la aplicación, se procederá llevar a cabo el plan de aseguramiento de calidad. Este plan se compone de tres procesos:

- Creación de pruebas.
- Ejecución y revisión de pruebas.

2.3.1. Creación de pruebas

A continuación se exponen los diferentes planes de calidad generados. Para la elaboración de los mismos se ha utilizado la documentación generada en el anterior apartado de extracción de conocimiento de la aplicación.

2.3.1.1. Planes de prueba Gestión_Películas

A continuación se exponen tres tipos de planes de pruebas, en función al enfoque con el que actuarán los correspondientes casos de prueba:

- Positivo: los casos de prueba actuarán de forma adecuada sobre la aplicación.
  - Insertar película.
  - Buscar película.
  - Puntuar película.

- Negativo: los casos de prueba actuarán de forma intencionadamente equivocada sobre la aplicación.
  - Insertar película.

- Carga: los casos de prueba provocarán comportamientos indeseados como consecuencia de carga extrema de datos sobre la aplicación.

- Insertar película.

2.3.1.1.1. Plan de pruebas 1 (positivo)

Planes-Prueba	
Información básica	
Identificador	Gestión_Películas-Positivo
Producto	Gestiona-España-1.0
Fecha creación	10-01-2009
Fecha revisión	16-02-2009
Modificado por	Ana Fernández Sánchez
Tiempo ejecución	6 horas.
Descripción	
El alcance del actual plan de pruebas estará relacionado con el módulo funcional de gestión de películas, el cual deberá verificar las siguientes funcionalidades: - Insertar película. - Buscar película. - Puntuar película.	
Caso(s) de uso relacionado(s)	
- CUOI-03 - CUOI-06 - CUOI-10	
Enfoque	
En los casos de prueba contenidos en el plan se interactuará con la aplicación de forma correcta.	
Pruebas automáticas	
- Caso de prueba: buscar_película 2	

2.3.1.1.1.1. Caso de pruebas insertar_película

Casos-Prueba	
Identificador	Gestión_Películas-Positivo-Insertar_Película-01
Nombre	Caso de prueba añadir película.
Persona encargada	Víctor Ayuso Busquets
Datos de partida/Estado inicial	o Estar en el apartado de películas.
Pasos	1- Del menú izquierdo, en el apartado de "Acciones sobre películas", pulsar en el icono de añadir una nueva película. 2- Del formulario expuesto, cumplimentar los diferentes campos como se expone: **Sinopsis:** "Resumen de la película". **Título:** "Mi película de prueba 1" **Nacionalidad:** "Española" **Año:** "2009" **Duración:** "120" **Tipo:** Pulsar sobre la ComboBox y seleccionar en el tipo "Comedia". **Dirección:** "Director 1" **Actriz Principal:** "Actor 1" **Actor Principal:** "Actriz 1" **Guión:** "Guionista 1" 3- Pulsar sobre el cartel de la película en blanco y a través del explorador que se abrirá, seleccionar cualquier archivo de imagen. 4- Abrir el explorador de Windows manualmente para seleccionar cualquier imagen y arrastrarla hasta la región en la que se encuentra el cartel de

	la película a insertar. 5- Pulsar en el icono de guardar.
Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. Al pasar el ratón por encima del icono de añadir una nueva película, deberá aparecer un borde recubriendo al mismo. 2. Una vez pulsado el icono de guardar la película, deberá ser sustituido el formulario de creación de la película por la vista ampliada de la nueva película introducida en la base de datos. 3. Cada campo que muestra la película deberá corresponder con los datos que se acaban de introducir, siendo el cartel de la película la última imagen que se ha introducido. 4. La vista de películas en miniatura localizada en la parte superior de la interfaz deberá haber sido actualizada con la nueva película añadida (con los datos correctamente almacenados).

2.3.1.1.1.2. Caso de pruebas buscar_película 1

Casos-Prueba	
Identificador	Gestión_Películas-Positivo-Buscar_Película-01
Nombre	Caso de prueba buscar película.
Persona encargada	Víctor Ayuso Busquets
Datos de partida/Estado inicial	o Estar en el apartado de películas. o Tener tres películas del tipo “Comedia” insertadas en la base de datos:

	Película1: **Nombre:** El otro lado de la cama **Año:** 2002 **Director:** Emilio Martínez-Lázaro **Actor:** Ernesto Alterio **Actriz:** Paz Vega Película2: **Nombre:** Días de fútbol **Año:** 2003 **Director:** David Serrano **Actor:** Ernesto Alterio **Actriz:** Natalia Verbeke Película3: **Nombre:** Mi película de prueba 1 **Año:** 2009 **Director:** Director 1 **Actor:** Actor 1 **Actriz:** Actriz 1 o La última búsqueda debe haberse realizado mostrando todos los campos posibles en la vista de películas en miniatura.
Pasos	1. Del menú izquierdo, en el apartado de “Acciones sobre búsquedas”, rellenar los siguientes campos según se expone: **Nombre:** dejar el campo vacío (“”). **Tipo:** Pulsar sobre la ComboBox y seleccionar en el tipo “Comedia”. **Año:** dejar el año vacío (“”). 2. Pulsar sobre el botón de buscar situado en la parte inferior del propio menú.

Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberán aparecer las tres películas mencionadas anteriormente en el apartado de *datos de partida.* 2. Por cada película deberán aparecer los siguientes campos: o Nombre de la película o Año o Director o Actores principales

2.3.1.1.1.3. Caso de pruebas buscar_película 2

Casos-Prueba	
Identificador	Gestión_Películas-Positivo-Buscar_Película-02
Nombre	Caso de prueba buscar película 2.
Persona encargada	Víctor Ayuso Busquets
Datos de partida/Estado inicial	o Estar en el apartado de películas.
Pasos	1. Del menú izquierdo, en el apartado de “Acciones sobre búsquedas”, rellenar los siguientes campos según se expone: **Nombre:** dejar el campo vacío (“”). **Tipo:** Pulsar sobre la ComboBox y seleccionar en el tipo “”. **Año:** “2010”. 2. Pulsar sobre el botón de buscar situado en la parte

	inferior del propio menú.
Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberá aparecer el siguiente texto: "No hay ninguna película con esa descripción".

2.3.1.1.1.4. Caso de pruebas buscar_película 3

Casos-Prueba	
Identificador	Gestión_Películas-Positivo-Buscar_Película-03
Nombre	Caso de prueba buscar película 3.
Persona encargada	Víctor Ayuso Busquets
Datos de partida/Estado inicial	o Estar en el apartado de películas. o Tener películas insertadas en la base de datos.
Pasos	1. Del menú izquierdo, en el apartado de "Acciones sobre búsquedas", rellenar los siguientes campos según se expone: **Nombre:** dejar el campo vacío (""). **Tipo:** Pulsar sobre la ComboBox y seleccionar en el tipo "". **Año:** dejar el año vacío (""). 2. Pulsar sobre el botón de buscar situado en la parte inferior del propio menú.
Tipo ejecución	Manual [X] Automática [ ]

Resultado esperado	1. Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberán aparecer todas las películas almacenadas en la base de datos.

2.3.1.1.1.5. Caso de pruebas buscar_película 4

Casos-Prueba	
Identificador	Gestión_Películas-Positivo-Buscar_Película-04
Nombre	Caso de prueba buscar película 4.
Persona encargada	Víctor Ayuso Busquets
Datos de partida/Estado inicial	o Estar en el apartado de películas. o Tener películas del año 2009 de tipo comedia insertadas en la base de datos.
Pasos	1. Del menú izquierdo, en el apartado de "Acciones sobre búsquedas", rellenar los siguientes campos según se expone: **Nombre:** dejar el campo vacío (""). **Tipo:** Pulsar sobre la ComboBox y seleccionar en el tipo "Comedia". **Año:** "2009". 2. Pulsar sobre el botón de buscar situado en la parte inferior del propio menú.
Tipo ejecución	Manual [ ] Automática [X]
Resultado esperado	1. En el fichero HTML generado, no deberá haber ningún tipo de error correspondiente al actual caso de prueba. 2. Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberán aparecer todas las

	películas del año 2009 y de tipo comedia.

2.3.1.1.1.5.1. Script Automático buscar_película 4

A continuación se muestra el script automático de dicho caso de prueba:

```
TextBox.SetValueAnimated("MenuPpal", Name, "textBoxNombre",
AutomationId, "");

ComboBox.SelectOrRemoveItem("comboBoxTipo", AutomationId, "Comedia",
Name, Select);

TextBox.SetValueAnimated("MenuPpal", Name, "textBoxAnyo",
AutomationId, "2009");

Button.Invoke("botonBuscar",AutomationId);
```

2.3.1.1.1.6. Caso de pruebas puntuar_película

Casos-Prueba	
Identificador	Gestión_Películas-Positivo-Puntuar_Película-01
Nombre	Caso de prueba puntuar película.
Persona encargada	Víctor Ayuso Busquets
Datos de partida/Estado inicial	o Tener una película maximizada
Pasos	1. En la parte superior de la película habrá 5 estrellas. Pulsar sobre las estrellas según se indica a continuación: a) Pulsar sobre la primera estrella. b) Situarse sobre la tercera estrella. c) Salir de la región de estrellas. d) Pulsar sobre la cuarta estrella. e) Pulsar sobre la segunda estrella. f) Pulsar sobre la quinta estrella. g) Situarse sobre la quinta estrella. h) Situarse sobre la cuarta estrella.

	i) Pulsar sobre la quinta estrella. j) Pulsar sobre la quinta estrella. 2. Cerrar la película en cuestión, para después volver a abrirla.
Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. En el paso b. deberán pintarse las tres primeras estrellas. 2. En el paso c. deberá pintarse únicamente la primera estrella. 3. En el paso g. deberán pintarse las cinco estrellas. 4. En el paso h. deberán pintarse las cuatro primeras estrellas. 5. Al cerrar la película deberá aparecer la vista por defecto, que se tratará de la lista del top-10 de películas. 6. Cuando se pretenda volver a abrir la película puntuada habrá que hacerlo a través de la lista de películas en miniatura. Cada vez que se pase el ratón por un elemento de dicha lista, deberá aparecer un borde de color rojo recubriendo la misma, e igualmente desapareciendo al salir de dicha región. 7. De las 5 estrellas que aparecen, como consecuencia de las anteriores puntuaciones, deberá haber 3 de ellas pintadas.

2.3.1.1.2. Plan de pruebas 2 (Negativo)

Planes-Prueba	
Información básica	
Identificador	Gestión_Películas-Negativo
Producto	Gestiona-España-1.0
Fecha creación	10-01-2009
Fecha revisión	16-02-2009
Modificado por	Ana Fernández Sánchez
Tiempo ejecución	2 horas.
Descripción	
El alcance del actual plan de pruebas estará relacionado con el módulo funcional de gestión de películas, el cual deberá verificar las siguientes funcionalidades: - Insertar película.	
Caso(s) de uso relacionado(s)	
- CUOI-03	
Enfoque	
En los casos de prueba contenidos en el plan se interactuará con la aplicación de forma intencionadamente equivocada.	
Pruebas automáticas	

2.3.1.1.2.1. Caso de pruebas insertar_película

Casos-Prueba	
Identificador	Gestión_Películas-Negativo-Insertar_Película-01
Nombre	Caso de prueba añadir película de forma equivocada.
Persona encargada	Luisa Aguado Bermellón
Datos de partida/Estado inicial	o Estar en el apartado de películas.
Pasos	1. Del menú izquierdo, en el apartado de "Acciones sobre películas", pulsar en el icono de añadir una nueva película. 2. Del formulario expuesto, cumplimentar los diferentes campos como se expone: **Sinopsis:** "Resumen de la película". **Título:** dejar el campo vacío (""). **Nacionalidad:** "Española" **Año:** "2009" **Duración:** "120" **Tipo:** Pulsar sobre la ComboBox y seleccionar en el tipo "Comedia". **Dirección:** "Director 1" **Actriz Principal:** "Actor 1" **Actor Principal:** "Actriz 1" **Guión:** "Guionista 1" 3. Abrir el explorador de Windows manualmente para seleccionar cualquier imagen y arrastrarla hasta la región en la que se encuentra el cartel de la película a insertar. 4. Pulsar en el icono de guardar.

Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. Al pasar el ratón por encima del icono de añadir una nueva película, deberá aparecer un borde recubriendo al mismo. 2. Una vez pulsado el icono de guardar la película, la aplicación deberá proporcionar información al usuario mencionando que el campo *título* de la película es obligatorio. 3. La película que se ha intentado insertar, en ningún momento debe haberse introducido en la base de datos.

2.3.1.1.3. Plan de pruebas 3 (Carga)

Planes-Prueba	
Información básica	
Identificador	Gestión_Películas-Carga
Producto	Gestiona-España-1.0
Fecha creación	10-01-2009
Fecha revisión	16-02-2009
Modificado por	Ana Fernández Sánchez
Tiempo ejecución	2 horas.
Descripción	

El alcance del actual plan de pruebas estará relacionado con el módulo funcional de gestión de películas, el cual deberá verificar las siguientes funcionalidades: - Insertar película.
Caso(s) de uso relacionado(s)
- CUOI-03
Enfoque
Los casos de prueba contenidos en el plan realizarán pruebas de carga a la hora de interactuar con la aplicación.
Pruebas automáticas
- Caso de prueba: insertar_película 2

2.3.1.1.3.1. Caso de pruebas insertar_película 1

Casos-Prueba	
Identificador	Gestión_Películas-Carga-Insertar_Película-01
Nombre	Caso de prueba insertar película con mucho contenido.
Persona encargada	Luisa Aguado Bermellón
Datos de partida/Estado inicial	o Estar en el apartado de películas.
Pasos	1. Del menú izquierdo, en el apartado de “Acciones sobre películas”, pulsar en el icono de añadir una nueva película. 2. Del formulario expuesto, cumplimentar los diferentes campos como se expone: **Sinopsis:** introducir un texto cualquiera que ocupe 20 líneas del control. **Título:** introducir un texto cualquiera que ocupe 75 caracteres. **Nacionalidad:** introducir un texto cualquiera que ocupe 75 caracteres.

	Año: introducir un texto cualquiera que ocupe 75 caracteres. **Duración:** introducir un texto cualquiera que ocupe 75 caracteres. **Tipo:** dejar el campo como está. **Dirección:** introducir un texto cualquiera que ocupe 75 caracteres. **Actriz Principal:** introducir un texto cualquiera que ocupe 75 caracteres. **Actor Principal:** introducir un texto cualquiera que ocupe 75 caracteres. **Guión:** introducir un texto cualquiera que ocupe 75 caracteres.
Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. Al pasar el ratón por encima del icono de añadir una nueva película, deberá aparecer un borde recubriendo al mismo. 2. Una vez cumplimentado el formulario, se deberá poder verificar correctamente cada campo rellenado.

2.3.1.1.3.2. Caso de pruebas insertar_película 2

Casos-Prueba	
Identificador	Gestión_Películas-Positivo-Insertar_Película-02
Nombre	Caso de prueba añadir película.
Persona encargada	Víctor Ayuso Busquets
Datos de partida/Estado inicial	o Estar en el apartado de películas.

Pasos	1- Ejecutar 5 veces los siguientes pasos: 2- Del menú izquierdo, en el apartado de “Acciones sobre películas”, pulsar en el icono de añadir una nueva película. 3- Del formulario expuesto, cumplimentar el formulario, poniendo como nombre de la película el que se muestra a continuación, y añadiendo un número secuencial mayor con cada iteración. **Sinopsis:** “Resumen de la película”. **Título:** “Mi película de prueba” **Nacionalidad:** “Española” **Año:** “2009” **Duración:** “120” **Tipo:** Pulsar sobre la ComboBox y seleccionar en el tipo “Comedia”. **Dirección:** “Director 1” **Actriz Principal:** “Actor 1” **Actor Principal:** “Actriz 1” **Guión:** “Guionista 1” 4- Pulsar en el icono de guardar.
Tipo ejecución	Manual [ ] Automática [X]
Resultado esperado	1. En el fichero HTML generado, no deberá haber ningún tipo de error correspondiente al actual caso de prueba. 2. Al final de la ejecución del script, deberá haber 5 películas nuevas en la lista de películas minimizadas, con sus respectivos campos.

2.3.1.1.3.2.1. Script Automático insertar_película 2

A continuación se muestra el script automático de dicho caso de prueba:

```
int i = 0;
while (i < 5)
{
Image.Invoke("MenuPpal", Name, "insertarImagen", AutomationId);
TextBox.SetValueAnimated("MenuPpal", Name, "modificarArgumento", AutomationId,
"Resumen de la película");
TextBox.SetValueAnimated("MenuPpal", Name, "modificarTitulo", AutomationId,
"Mi pelicula de prueba"+i);
TextBox.SetValueAnimated("MenuPpal", Name, "modificarPais", AutomationId,
"Española");
TextBox.SetValueAnimated("MenuPpal", Name, "modificarAnyo", AutomationId,
"2009");
TextBox.SetValueAnimated("MenuPpal", Name, "modificarDuracion", AutomationId,
"120");
ComboBox.SelectOrRemoveItem("modificarTipo", AutomationId, "Comedia", Name,
Select);
TextBox.SetValueAnimated("MenuPpal", Name, "modificarDirector", AutomationId,
"Director 1");
TextBox.SetValueAnimated("MenuPpal", Name, "modificarActorFem", AutomationId,
"Actriz 1");
TextBox.SetValueAnimated("MenuPpal", Name, "modificarGuionista", AutomationId,
"Guionista 1");
TextBox.SetValueAnimated("MenuPpal", Name, "modificarActorMasc", AutomationId,
"Actor 1");
Image.Invoke("MenuPpal", Name, "guardar", AutomationId);
i++;
}
```

2.3.1.2. Planes de prueba Gestión_Visualización

A continuación se expone el plan de pruebas generado para probar la funcionalidad del módulo funcional *Gestión de Visualización*.

- Positivo: los casos de prueba actuarán de forma adecuada sobre la aplicación.
  - Vistas_Lista_Miniatura.
  - Diferentes vistas de lista del TOP 10 de películas

2.3.1.2.1. Plan de pruebas (Positivo)

Planes-Prueba	
Información básica	
Identificador	Gestión_Visualización-Positivo
Producto	Gestiona-España-1.0
Fecha creación	10-01-2009
Fecha revisión	16-02-2009
Modificado por	Ana Fernández Sánchez
Tiempo ejecución	3 horas.
Descripción	
El alcance del actual plan de pruebas será el relacionado con el módulo funcional de gestión de visualización, el cual deberá verificar las siguientes funcionalidades: - Vistas_Lista_Miniatura. - Diferentes vistas de lista del TOP 10 de películas	
Caso(s) de uso relacionado(s)	
- CUOI-07 - CUOI-02	
Enfoque	
En los casos de prueba contenidos en el plan se interactuará con la aplicación de forma correcta.	
Pruebas automáticas	

2.3.1.2.1.1. Caso de pruebas vistas_lista_miniatura

Casos-Prueba	
Identificador	Gestión_Visualización-Positivo-Vistas_Lista_Miniatura-01
Nombre	Caso de prueba cambiar vista de la lista de películas minimizadas.
Persona encargada	Luisa Aguado Bermellón
Datos de partida/Estado inicial	o Estar en el apartado de películas. o Que la última búsqueda haya encontrado alguna película.
Pasos	1. Del menú izquierdo, en el apartado de "Acciones sobre búsquedas", expandir el control *Expander*. 2. Desmarcar todos los *CheckBox* exceptuando el de *Nombre de la película* y el de *año de la película*. 3. Pulsar el botón que aparece más abajo.
Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. El control Expander deberá permanecer expandido, y cuyas CheckBox de *nombre de la película y año de la película* marcados. 2. Cada película que aparece en la lista de películas minimizadas deberá mostrar únicamente los campos nombre y año.

2.3.1.2.1.2. Caso de pruebas vistas_lista_top10

Casos-Prueba	
Identificador	Gestión_Visualización-Positivo-Vistas_Lista_TOP_10-01
Nombre	Caso de prueba cambiar vista de la lista del TOP-10 de películas.
Persona encargada	Luisa Aguado Bermellón
Datos de partida/Estado inicial	o Estar en el apartado de películas. o Tener visible el TOP-10 de películas. o El puntero del *Slider* se debe encontrar en la región 1.
Pasos	1. Pulsar (sin soltar) sobre el puntero del control *Slider* que aparece sobre la lista. 2. Arrastrar el control hasta la región 3.
Tipo ejecución	Manual [X] Automática [ ]
Resultado esperado	1. Cada elemento de la lista debe pasar de mostrar los campos *nombre y argumento*, a mostrar *nombre, año, director, guionista y argumento.* 2. El número de la región 1 debe terminar en color blanco, quedando el de la correspondiente región 3 en color rojo.

2.3.2. Ejecución y revisión de pruebas

En el momento en el que todos los casos de prueba de los correspondientes planes de prueba definidos hayan sido completados, será momento de la ejecución de los mismos.

Para poder llevar a cabo dichas pruebas, primeramente deberemos tener preparado el entorno adecuado, el cual deberá ser el más similar al del futuro cliente, para certificar que los resultados obtenidos lo serán sobre esas condiciones del entorno (S.O, cantidad de memoria…).

2.3.2.1. Preparación del entorno

Una vez que todos los casos de prueba están definidos, habrá que preparar el entorno para llevar a cabo dichas pruebas. La documentación generada durante esta tarea es la siguiente:

Entorno-Pruebas			
Especificaciones HW			
Procesador	Pentium 4		
Velocidad	1.4 GHz		
Memoria RAM	1 GB		
Especificaciones SW			
Sistema operativo	Microsoft Windows Vista		
Resolución pantalla	1280 x 800		
Software necesario	Microsoft .Net Framework	**Versión**	3.5

2.3.2.2. Ejecución y revisión de los casos de prueba

Una vez que dispongamos del entorno necesario para poder ejecutar la aplicación objeto de QA, será el momento de ejecutar los casos de prueba creados en anteriores tareas.

Como consecuencia de las ejecuciones de los casos de prueba definidos, se ha obtenido los siguientes resultados:

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Positivo-Insertar_Película-01
Resultado esperado	1. Al pasar el ratón por encima del icono de añadir una nueva película, deberá aparecer un borde recubriendo al mismo. 2. Una vez pulsado el icono de guardar la película, deberá ser sustituido el formulario de creación de la película por la vista ampliada de la nueva película introducida en la base de datos. 3. Cada campo que muestra la película deberá corresponder con los datos que se acaban de introducir, siendo el cartel de la película la última imagen que se ha introducido. 1. La vista de películas en miniatura localizada en la parte superior de la interfaz deberá haber sido actualizada con la nueva película añadida (con los datos correctamente almacenados).
Resultado obtenido	1. Al pasar el ratón por encima del icono de añadir una nueva película aparece un borde recubriendo al mismo. 2. Una vez pulsado el icono de guardar la película,

	aparece la vista ampliada de la película introducida. 3. Todos los datos son los que se han introducido en el formulario. 4. La película aparece en la lista de películas minimizadas.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Positivo-Buscar_Película-01
Resultado esperado	1. Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberán aparecer las tres películas mencionadas anteriormente en el apartado de datos de partida. 2. Por cada película deberán aparecer los siguientes campos: o Nombre de la película o Año o Director o Actores principales
Resultado obtenido	1. Como consecuencia de llevar a cabo la búsqueda, en la vista de películas en miniatura aparecerán las siguientes películas: o El otro lado de la cama. o Días de fútbol. o Mi película de prueba 1. 2. Por cada película se obtiene el *nombre de la película*, *año*, *director* y *actores principales*.

Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Positivo-Buscar_Película-02
Resultado esperado	Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberá aparecer el siguiente texto: “No hay ninguna película con esa descripción”.
Resultado obtenido	Al realizar la búsqueda no hay ninguna película que cumpla con las especificaciones, mostrándose un comentario correspondiente.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Positivo-Buscar_Película-03
Resultado esperado	Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberán aparecer todas las películas almacenadas en la base de datos.
Resultado obtenido	Al realizar la búsqueda, aparecen todas las películas que hay en la base de datos.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Positivo-Buscar_Película-04
Resultado esperado	1. En el fichero HTML generado, no deberá haber ningún tipo de error correspondiente al actual caso de prueba. 2. Una vez pulsado el botón de buscar, en la vista de películas en miniatura deberán aparecer todas las películas del año 2009 y de tipo comedia.
Resultado obtenido	1. El caso de prueba respecto al fichero HTML generado es satisfactorio. 2. Al realizar la búsqueda, aparecen todas las películas del año 2009 y de tipo comedia.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Positivo-Puntuar_Película-01
Resultado esperado	1. En el paso b. deberán pintarse las tres primeras estrellas. 2. En el paso c. deberá pintarse únicamente la primera estrella. 3. En el paso g. deberán pintarse las cinco estrellas. 4. En el paso h. deberán pintarse las cuatro primeras estrellas. 5. Al cerrar la película deberá aparecer la vista por defecto, que se tratará de la lista del top-10 de películas.

	6. Cuando se pretenda volver a abrir la película puntuada habrá que hacerlo a través de la lista de películas en miniatura. Cada vez que se pase el ratón por un elemento de dicha lista, deberá aparecer un borde de color rojo recubriendo la misma, e igualmente desapareciendo al salir de dicha región. 7. De las 5 estrellas que aparecen, como consecuencia de las anteriores puntuaciones, deberá haber 3 de ellas pintadas.
Resultado obtenido	1. Al situarse sobre la tercera estrella del paso b. se marcan las tres primeras. 2. Al salir de la región del paso c. se queda marcada la primera. 3. Al situarse sobre la quinta estrella del paso g. se quedan marcadas todas las estrellas. 4. Al situarse sobre la cuarta estrella del paso h. se quedan marcadas las cuatro primeras estrellas. 5. Al cerrar la película aparece la vista del top-10 de películas. 6. Al recorrer la lista de miniatura de películas con motivo de abrir de nuevo la que acabamos de puntuar, se van marcando y desmarcando los bordes de cada una de ellas cuando pasamos el ratón por encima. 7. Al abrir de nuevo la película, esta tiene una puntuación de 3 estrellas.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Negativo-Insertar_Película-01
Resultado esperado	1. Al pasar el ratón por encima del icono de añadir una nueva película, deberá aparecer un borde recubriendo al mismo. 2. Una vez pulsado el icono de guardar la película, la aplicación deberá proporcionar información al usuario mencionando que el campo *título* de la película es obligatorio. 3. La película que se ha intentado insertar, en ningún momento debe haberse introducido en la base de datos.
Resultado obtenido	1. Al pasar el ratón por encima del icono de añadir una nueva película aparece un borde recubriendo al mismo. 2. Una vez pulsado el icono de guardar la película, aparece la vista ampliada de la película introducida. 3. Al guardar la película, todos los datos son los que se han introducido en el formulario. 4. La película aparece en la lista de películas minimizadas.
Conclusiones	Los resultados esperados no se han cumplido, esto es debido a que un requisito de usuario era que el nombre de una película debe tener algún valor. En este caso nos ha dejado introducir una película sin título.
Resolución	**Correcto** [ ] **Incorrecto** [X]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Carga-Insertar_Película-01
Resultado esperado	1. Al pasar el ratón por encima del icono de añadir una nueva película, deberá aparecer un borde recubriendo al mismo. 2. Una vez cumplimentado el formulario, se deberá poder verificar correctamente cada campo rellenado.
Resultado obtenido	1. Al pasar el ratón por encima del icono de añadir una nueva película aparece un borde recubriendo al mismo. 2. Los siguientes campos del formulario hacen que el contenido del mismo se desplace sin existir ninguna manera de acceder a los controles que desaparecen del espacio visible: o Sinopsis o Nacionalidad o Año o Duración o Dirección o Actor y actriz principales o Guión
Conclusiones	Los resultados esperados no se han cumplido, esto es debido a que en el momento en el que se introducen muchos datos en un determinado campo, se desplaza el resto de campos sin existir ninguna posibilidad de acceder a los mismos.
Resolución	**Correcto** [ ] **Incorrecto** [X]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Películas-Carga-Insertar_Película-02
Resultado esperado	1. En el fichero HTML generado, no deberá haber ningún tipo de error correspondiente al actual caso de prueba. 1. Al final de la ejecución del script, deberá haber 5 películas nuevas en la lista de películas minimizadas, con sus respectivos campos.
Resultado obtenido	1. El caso de prueba respecto al fichero HTML generado es satisfactorio. 2. Al realizar las inserciones, aparecen todas las películas en la lista de películas minimizadas.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Visualización-Positivo-Vistas_Lista_Miniatura-01
Resultado esperado	Cada película que aparece en la lista de películas minimizadas deberá mostrar únicamente los campos nombre y año.
Resultado obtenido	1. Al realizar la búsqueda, el control Expander permanece expandido. 2. Los CheckBox correspondientes al *nombre de la película* y el *año* permanecen marcados. 3. Todas las películas que aparecen en la lista de películas minimizadas muestran únicamente los campos *nombre de la* película y el año, además del

	cartel.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Resultado-Casos-Prueba	
Id. Caso de prueba	Gestión_Visualización-Positivo-Vistas_Lista_TOP_10-01
Resultado esperado	1. Cada elemento de la lista debe pasar de mostrar los campos *nombre y argumento*, a mostrar *nombre, año, director, guionista y argumento.* 2. El número de la región 1 debe terminar en color blanco, quedando el de la correspondiente región 3 en color rojo.
Resultado obtenido	1. Al mover el puntero del control Slider a la tercera región se ha cambiado la vista de la lista, la cual muestra: o Nombre de la película o Año o Director o Guionista o Argumento 2. El número de la región 1 cambia a color blanco y el de la región 3 se pone en rojo.
Conclusiones	Todos los resultados esperados se han cumplido, por lo que el caso de prueba es satisfactorio.
Resolución	**Correcto** [X] **Incorrecto** [ ]

Finalmente se reportarán los casos de prueba que no hayan sido satisfactorios al equipo de desarrollo. Estos casos de prueba serán:

LISTADO-BUGS	
ID. PLAN PRUEBAS	**Gestión_Películas-Negativo**
ID. CASO PRUEBA	Gestión_Películas-Negativo-Insertar_Película-01
ID. PLAN PRUEBAS	**Gestión_Películas-Carga**
ID. CASO PRUEBA	Gestión_Películas-Carga-Insertar_Película-01

3. Planificación y presupuesto de las pruebas

A continuación se expone el presupuesto y planificación necesarios para llevar a cabo la fase de pruebas sobre la interfaz de usuario:

3.1. Desglose por fases

A continuación se presentan las horas totales de cada una de las fases seguidas en la puesta en marcha de la metodología de pruebas.

Fase\Horas	Horas totales
Estudio de viabilidad	10
Extracción de conocimiento y generación de documentación preliminar	38
Plan de aseguramiento de calidad	22
Total horas	70

Tabla 44: horas/fase pruebas

3.2. Planificación

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Puesta en marcha metodología**	**14 días**	**lun 19/01/09**	**jue 29/01/09**
2	Estudio Viabilidad	2 días	lun 19/01/09	mar 20/01/09
3	**ECGD**	**7,6 días**	**mar 20/01/09**	**lun 26/01/09**
4	Análisis módulos funcionales	1,6 días	mar 20/01/09	mié 21/01/09
5	Estudio controles	6 días	mié 21/01/09	lun 26/01/09
6	**Plan aseguram. Calidad**	**4,4 días**	**mar 27/01/09**	**jue 29/01/09**
7	Creación planes/casos prueba	1,4 días	mar 27/01/09	mar 27/01/09
8	Ejecución casos prueba	3 días	mar 27/01/09	jue 29/01/09

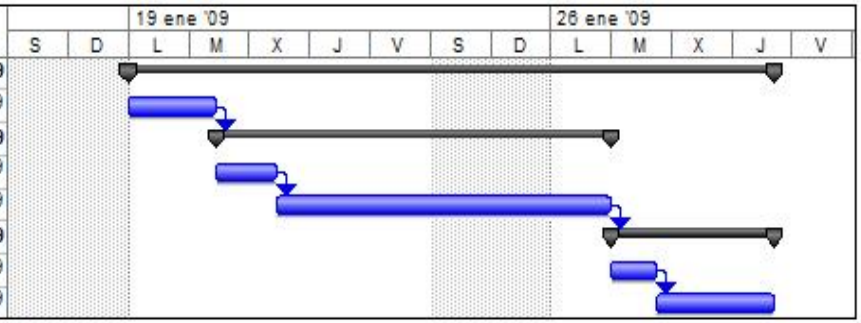


Ilustración 49: Gantt pruebas

3.3. Salarios por categoría

Para conocer el gasto de la empresa en relación al trabajo de sus empleados o personal que conforma la misma es necesario conocer los salarios de los trabajadores.

Los salarios medios se definen en la siguiente tabla:

Cargo	Sueldo Bruto Año	Sueldo Bruto mes	Coste Hora/empleado
JEC	50.000 €	4.166 €	26,041 €
Analista de sistemas	30.000 €	2.500 €	15,625 €
Analista de pruebas	27.000 €	2.250 €	14,062 €
Tester	15.000 €	1.250 €	7,812 €

Tabla 45: Coste salarios puesto/hora pruebas

A continuación, tras conocer los salarios de los empleados hay que valorar cuantas horas del proyecto va a dedicar cada uno de los empleados de la empresa, en relación a las distintas fases del proceso, incluyendo los costes asociados:

Fase	JEC (horas)	Analista de sistemas (horas)	Analista de pruebas (horas)	Tester (horas)	Total (€)
Estudio de viabilidad	3	7			187,498
Extracción de conocimiento y generación de documentación preliminar	8	16	7	7	611,446
Plan de aseguramiento de calidad	2	2	8	10	273,948
TOTAL	**13*26,041= 338,533 €**	**25*15,625= 390,625 €**	**15*14,062= 210,93 €**	**17*7,812= 132,804 €**	**1.072,892**

Tabla 46: Coste salarios de personal/fase pruebas

3.4. Gastos de personal imputables al proyecto

A partir de la tabla anterior es posible sacar la relación de costes de los empleados del proyecto en relación a las horas totales que va a dedicar cada uno de ellos al mismo:

Empleado	Horas	Coste
JEC	13 h	13*26,041=338,533 €
Analista de sistemas	25 h	25*15,625=390,625 €
Analista de pruebas	15 h	15*14,062=210,93 €
Tester	8,5 h (x2)	8,5*2*7,812=132,804 €
TOTAL	**70 h**	**1.072,892 €**

Tabla 47: Coste horas/empleados

3.5. Gastos en materiales

En este apartado es necesario incluir todo el material necesario para llevar a cabo las pruebas sobre la aplicación. El material necesario estará formado principalmente por los equipos de trabajo del personal de pruebas de la empresa. En la siguiente tabla se resume el total de gastos de materiales:

Material	Cantidad	Coste
Ordenador Portatil Dell Inspiron 64 bits 1 GB RAM 60 GB Disco Duro	1	499 €
Sistema Operativo Windows Vista Profesional	1	300 €
TOTAL	-	**799 €**

Tabla 48. Gasto Materiales pruebas

3.6. Resumen del presupuesto

En esta tabla se especifican todos los gastos obtenidos anteriormente para la realización del resumen del presupuesto:

Gasto	Coste
Empleados	1.072,892 €
Gastos en materiales	799 €
SUBTOTAL	1.871,892 €
IVA (16%)	299,502 €
TOTAL	**2.171,394 €**

Tabla 49: Resumen Presupuesto pruebas

El presupuesto final de la puesta en marcha de las pruebas asciende a un TOTAL de **2.171,394 €** (DOS MIL CIENTO SETENTA Y UNO, CON TRESCIENTOS NOVENTA Y CUATRO EUROS) **IVA INC.**

VI

Presupuesto final

VI. Presupuesto final

A continuación se expone el presupuesto y planificación globales del proyecto de final de carrera:

1. Desglose por fases

A continuación se presentan las horas totales de cada una de las fases seguidas para llevar a cabo el proyecto.

Fase \ Horas	Horas totales
Creación de la metodología de QA.	150
Creación de la aplicación WPF objeto de QA.	230
Puesta en marcha de la metodología.	70
Documentación general	40
Total horas	490

Tabla 50: fase/horas PFC

1.1. Planificación

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Proyecto Fin de Carrera**	**98 días**	**vie 14/11/08**	**lun 09/02/09**
2	Creación metodología de calidad	30 días	vie 14/11/08	mié 10/12/08
3	Creación aplicación WPF	46 días	mié 10/12/08	mar 20/01/09
4	Fase de pruebas	14 días	mar 20/01/09	lun 02/02/09
5	Documentación	8 días	lun 02/02/09	lun 09/02/09

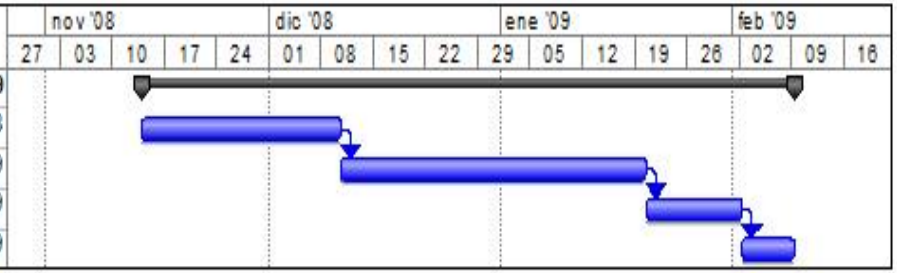


Ilustración 50: Gantt PFC

2. Resumen del presupuesto

El presupuesto final se elabora teniendo en cuenta las diferentes partes que conforman el proyecto fin de carrera:

ACTIVIDAD	TOTAL
Creación de la metodología de QA.	2.256,674 € IVA INC (16%).
Creación de la aplicación WPF objeto de QA.	7.403,99 € IVA INC (16%).
Puesta en marcha de la metodología.	2.171,394 € IVA INC. (16%)
Documentación	725 € IVA INC. (16%)
Total	12.557 €

Tabla 51: Presupuesto PFC

El presupuesto final del proyecto del final de carrera asciende a un TOTAL de **12.557 €** (DOCE MIL QUINIENTOS CINCUENTA Y SIETE EUROS) **IVA INC.**

VII

Conclusiones y trabajos futuros

VII. Conclusiones y trabajos futuros

1. Problemas encontrados

Durante el desarrollo del proyecto me he encontrado con múltiples problemas, en la mayoría de los casos referentes a la creación de la metodología de calidad. Esto ha sido debido a la falta de experiencia que tenía en el desarrollo de documentos formales de este tipo, unido a que tampoco he podido encontrar información de gran utilidad que aportara conocimientos como por ejemplo, factores importantes a la hora de crear una metodología.

Por otra parte, he tenido que aprender todo lo relacionado con aplicaciones de tecnología .NET puesto que no tenía ningún tipo de experiencia con dicha plataforma, debiendo invertir gran cantidad de tiempo adquirir conocimientos de este tipo.

2. Conclusiones

Para terminar, cabe destacar que se han cumplido todos y cada uno de los objetivos planteados desde un principio. Con la puesta en marcha de la metodología sobre la aplicación se ha demostrado la dificultad de que después de la fase de implementación un producto permanezca sin defectos, habiendo encontrado varios. Dichos defectos serán conocidos por parte del equipo de desarrollo, por lo que presumiblemente en un futuro podrán ser arreglados.

Al haber terminado la puesta en marcha de la metodología, no podremos afirmar que la interfaz de la aplicación no contiene errores, pero sí podemos afirmar (y consta formalmente) que para los casos de prueba que se han ejecutado de forma controlada la aplicación responde adecuadamente, certificando el correcto funcionamiento de dichas partes.

Esto último mencionado es algo de vital importancia, sobretodo en grandes proyectos. Esto es debido a que en muchas ocasiones, en escenarios en los que no se utiliza ningún tipo de metodología, realmente no se tiene constancia de qué cosas se han probado y si estas funcionan, no pudiendo hacer estimaciones sobre la calidad global de que dispone una determinada aplicación (madurez) en términos cuantitativos.

3. Observaciones personales

Cabe mencionar que este proyecto de fin de carrera me ha sido de gran utilidad para realmente tener una noción más cercana sobre el tema de calidad del software. En general durante la carrera, no hay ninguna asignatura que dedique un gran bloque a la calidad, al menos en cuanto a los procesos a seguir para probar un determinado software. Por ello considero que he aprendido algo nuevo, debiendo de tenerse este tema más en cuenta en futuros planes de estudios.

El hecho de tratar un tema tan importante como lo es el aseguramiento de calidad, en el que se ha definido una metodología para llevar a cabo pruebas sobre interfaces de un determinado tipo de aplicaciones, obliga prácticamente a desarrollar una aplicación de este tipo para llevar a cabo el proceso mencionado en la metodología planteada. Gracias a esto último mencionado, también he adquirido abundantes conocimientos sobre la plataforma .NET, y en el desarrollo de aplicaciones visuales de última generación, algo que me ha gustado enormemente.

A rasgos generales estoy muy satisfecho respecto a las cosas que he aportado en el proyecto de fin de carrera, pero sobre todo por las cosas que me llevo conmigo, habiendo tenido además un tutor siempre dispuesto a ayudar y que me ha sabido guiar por el buen camino en todo momento.

4. Líneas futuras

Como **líneas futuras** se podría plantear, desde el punto de vista de la metodología desarrollada, la creación de diferentes métricas para cerrar el ciclo de calidad, como por ejemplo:

- Para indicar la calidad del producto.
- Para evaluar la productividad de la gente que desarrolla el producto.
- Par evaluar los beneficios en términos de productividad y de calidad, derivados del uso de la metodología.

Este planteamiento sería muy interesante para que además de poder obtener posibles las deficiencias de la aplicación, podamos cuantificar de alguna forma elementos que pueden ser interesantes de analizar.

VIII

Bibliografía y Referencias

VIII. Bibliografía y Referencias

Dustin, E., Rashka, J., & Paul, J. (1999). Automated Software Testing "Introduction, management and performance". Addison-Wesley

Fewster, M., & Graham, D. (1999). Software Test Automation "Effective use of test execution tools". Addison-Wesley

IRM (2005). "Metodologías, enfoques y técnicas". Obtenido de http://www.geocities.com/autogestion/metodologia/metodologia.html

Kaner, C., Falk, J., & Quoc, H. (1993). Testing Computer Software (2ª Edición). International Thompson Computer Press.

Cal-Métrica3. Página WEB CSAE. Obtenido de http://www.csae.map.es/csi/metrica3/calidad.pdf

Microsoft Corporation (2007). "Introducción WPF". Sitio WEB de MSDN. Obtenido de http://msdn.microsoft.com/es-es/library/ms742119.aspx

Microsoft Corporation (2007). "Controles WPF por categoría". Sitio WEB de MSDN. Obtenido de http://msdn.microsoft.com/es-es/library/ms754204.aspx

Microsoft Corporation. (2007). "XAML Overview". Sitio WEB de MDSN. Obtenido de http://msdn2.microsoft.com/en-us/library/ms752059.aspx

Microsoft Corporation. (2008). Sitio WEB de MSDN. Obtenido de http://msdn.microsoft.com/en-us/netframework/default.aspx

Montesinos, O. (2006). "Framework 3.0, WPF y XAML". Obtenido de http://oscarmontesinos.blogspot.com/2006/11/framework-30-wpf-xaml.html

Sierra, M. (2008). ¿Cómo aplicar procesos de calidad al desarrollo de software? Obtenido de *CIC-Calidad en el desarrollo de software.pdf*

Sierra, M. (2008). Calidad en la interfaz de usuario. Obtenido de http://geeks.ms/blogs/msierra/archive/2008/10/30/calidad-en-la-interfaz-de-usuario.aspx

Villarroel, R. H. (2007). Roles en el desarrollo de software. Obtenido de http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1/Roles_desarrollo_software.pdf

Wikipedia. (2006). Sitio WEB de Wikipedia. Obtenido de http://es.wikipedia.org/wiki/XAML

Wikipedia (2008). "Microsoft .NET. Obtenido de http://es.wikipedia.org/wiki/.NET_de_Microsoft